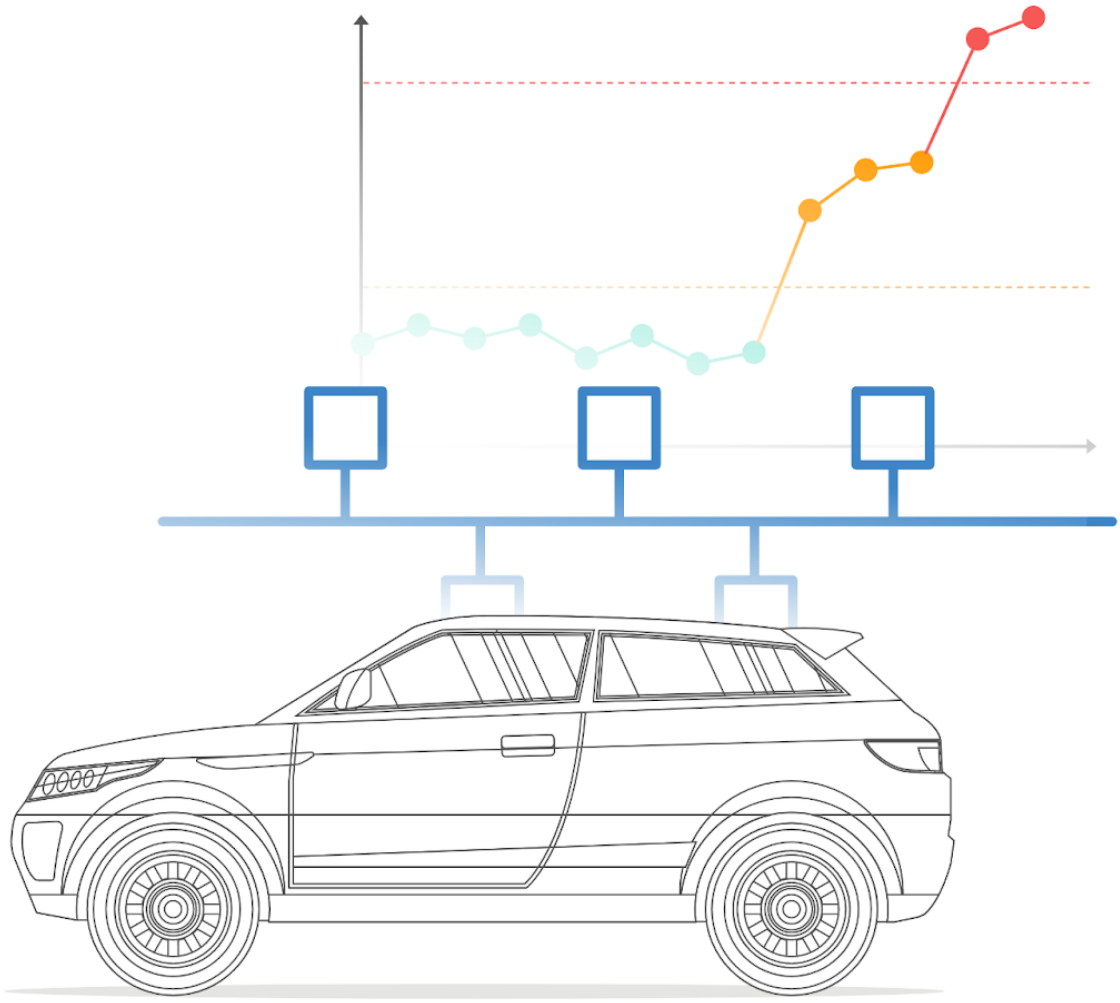


CAN bus

the ultimate guide

CAN | J1939 | OBD2 | UDS | CANopen | CAN FD | LIN | DBC | CAN Errors

NMEA 2000 | ISOBUS | CCP/XCP on CAN

CSS Electronics | www.csselectronics.com | contact@csselectronics.com
Last modified: June 21, 2023

Table of Contents

CANedge - 2 x CAN/LIN Bus Data Logger

Interested in CAN bus and related protocols? Then make sure to check out our CANedge series as well!

The CANedge is a 2 x CAN/LIN data logger with an 8-32 GB industrial SD card. The series offers optional GPS/IMU, WiFi and/or 3G/4G - ideal for OEM diagnostics, vehicle fleet telematics, predictive maintenance and more:

PLUG & PLAY
Log CAN data out-the-box. Standalone. Bit-rate auto detect. Power safe

PRO SPECS
Extractable 8-32 GB SD. 2xCAN/LIN. CAN FD. Zero data loss. 50 Âµs RTC. Error frames. MF4

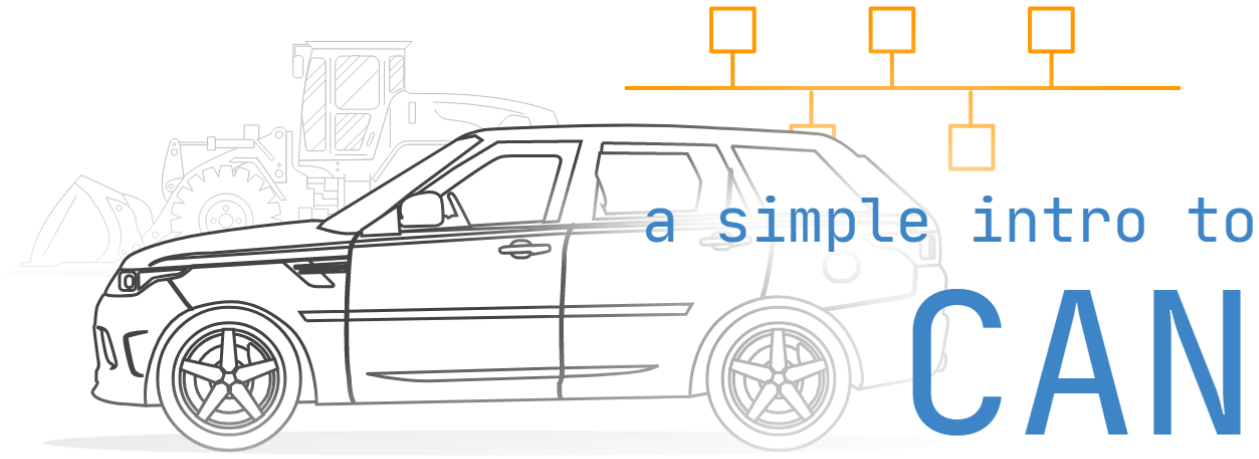
COMPACT
Only 8 x 5 x 2 CM. 100G. Robust alu enclosure. 5+ LEDs. 5V power out (CH2)

WIFI/LTE
Push data via WiFi or 3G/4G to your server. E2E security. OTA updates

GPS + 3D IMU
Built-in GPS/IMU. 3x accuracy via sensor fusion. Position, speed, distance & more

INTEROPERABLE
Free open source software/APIs. MF4 to ASC/CSV. DBC decoding. Dashboards

Check out the **5 min CANedge intro video**



CAN Bus Explained - A Simple Intro

Need a simple, practical intro to CAN bus? In this tutorial we explain the Controller Area Network (CAN bus) 'for dummies' incl. message interpretation, CAN logging - and the link to OBD2, J1939 and CANopen.

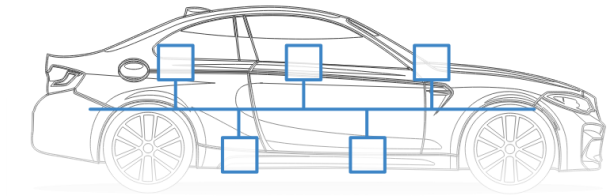
Read on to learn why this has become the #1 guide on CAN bus.

What is CAN bus?

Your car is like a human body:

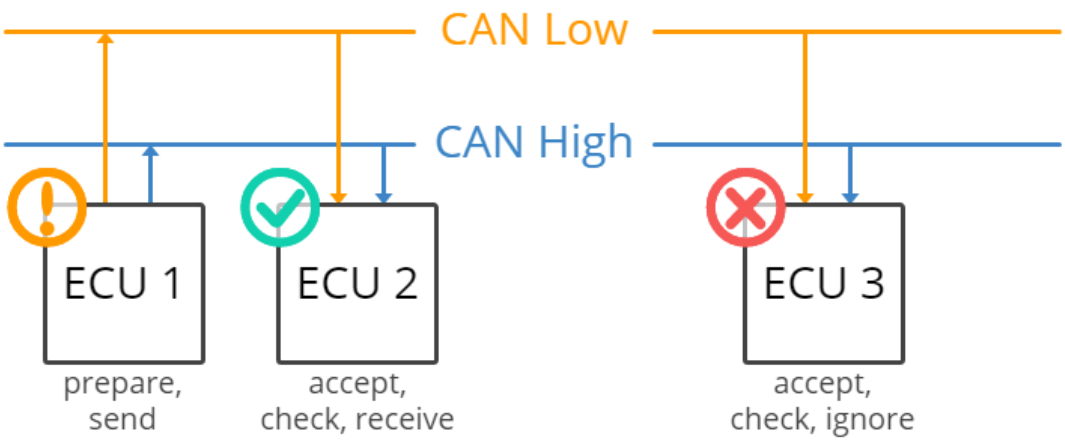
The Controller Area Network (CAN bus) is the nervous system, enabling communication.

In turn, 'nodes' or 'electronic control units' (ECUs) are like parts of the body, interconnected via the CAN bus. Information sensed by one part can be shared with another.



So what is an ECU?

In an automotive CAN bus system, ECUs can e.g. be the engine control unit, airbags, audio system etc. A modern car may have up to 70 ECUs - and each of them may have information that needs to be shared with other parts of the network.



This is where the CAN standard comes in handy:

The CAN bus system enables each ECU to communicate with all other ECUs - without complex dedicated wiring.

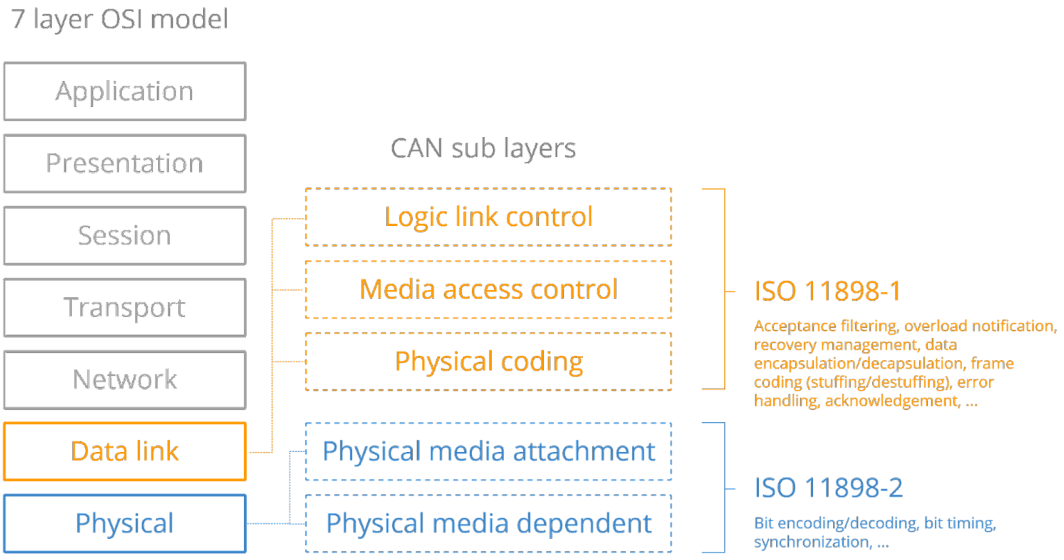
Specifically, an ECU can prepare and broadcast information (e.g. sensor data) via the CAN bus (consisting of two wires, CAN low and CAN high). The broadcasted data is accepted by all other ECUs on the CAN network - and each ECU can then check the data and decide whether to receive or ignore it.

CAN bus physical & data link layer (OSI)

In more technical terms, the controller area network is described by a data link layer and physical layer. In the case of high speed CAN, ISO 11898-1 describes the data link layer, while ISO 11898-2 describes the physical layer. The role of CAN is often presented in the 7 layer OSI model as per the illustration.

The CAN bus physical layer defines things like cable types, electrical signal levels, node requirements, cable impedance etc. For example, ISO 11898-2 dictates a number of things, including below:

- Baud rate: CAN nodes must be connected via a two wire bus with baud rates up to 1 Mbit/s (Classical CAN) or 5 Mbit/s (CAN FD)
- Cable length: Maximal CAN cable lengths should be between 500 meters (125 kbit/s) and 40 meters (1 Mbit/s)
- Termination: The CAN bus must be properly terminated using a 120 Ohms CAN bus termination resistor at each end of the bus



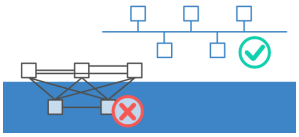
High speed CAN, low speed CAN, LIN bus, ...

In the context of automotive vehicle networks, you'll often encounter a number of different types of network types. Below we provide a very brief outline:

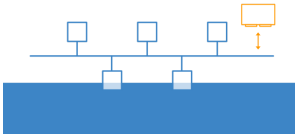
- **High speed CAN bus:** The focus of this article is on high speed CAN bus (ISO 11898). It is by far the most popular CAN standard for the physical layer, supporting bit rates from 40 kbit/s to 1 Mbit/s (Classical CAN). It provides simple cabling and is used in practically all automotive applications today. It also serves as the basis for several higher layer protocols such as OBD2, J1939, NMEA 2000, CANopen etc. The second generation of CAN is referred to as CAN FD (CAN with Flexible Data-rate)
- **Low speed CAN bus:** This standard enables bit rates from 40 kbit/s to 125 kbit/s and allows the CAN bus commmunication to continue even if there is a fault on one of the two wires - hence it is also referred to as 'fault tolerant CAN'. In this system, each CAN node has it's own CAN termination
- **LIN bus:** LIN bus is a lower cost supplement to CAN bus networks, with less harness and cheaper nodes. LIN bus clusters typically consist of a LIN master acting as gateway and up to 16 slave nodes. Typical use cases include e.g. non-critical vehicle functions like aircondition, door functionality etc. - for details see our LIN bus intro or LIN bus data logger article
- **Automotive ethernet:** This is increasingly being rolled out in the automotive sector to support the high bandwidth requirements of ADAS (Advanced Driver Assistance Systems), infotainment systems, cameras etc. Automotive ethernet offers much higher data transfer rates vs. CAN bus, but lacks some of the safety/performance features of Classical CAN and CAN FD. Most likely, the coming years will see both automotive ethernet, CAN FD and CAN XL being used in new automotive and industrial development

Top 4 benefits of CAN bus

The CAN bus standard is used in practically all vehicles and many machines due to below key benefits:

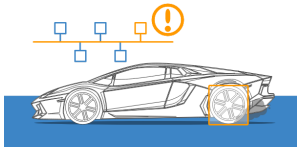


Simple & low cost
ECUs communicate via a single CAN system instead of via direct complex analogue signal lines - reducing errors, weight, wiring and costs



Fully centralized
The CAN bus provides 'one point-of-entry' to communicate with all network ECUs - enabling central diagnostics, data logging and configuration

Extremely robust
The system is robust towards electric disturbances and electromagnetic interference - ideal for safety critical applications (e.g. vehicles)

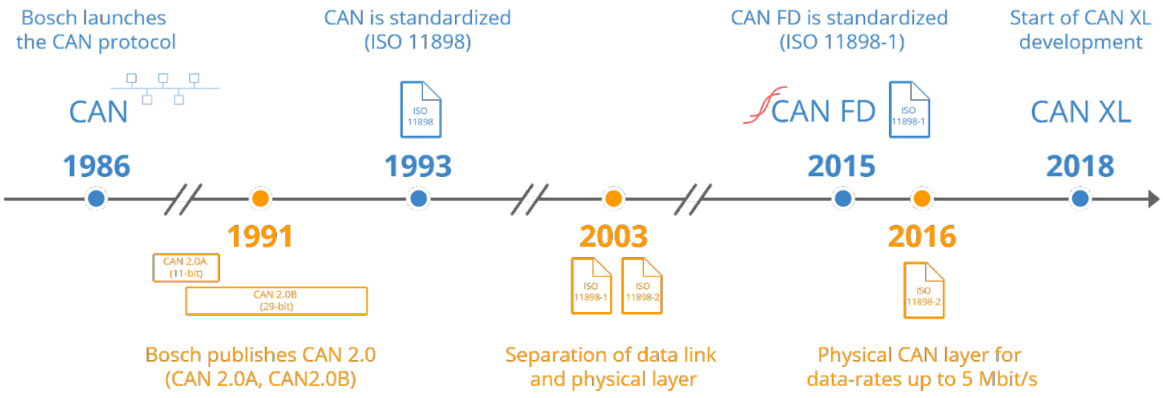


Efficient
CAN frames are prioritized by ID so that top priority data gets immediate bus access, without causing interruption of other frames

The CAN bus history in short

- Pre CAN: Car ECUs relied on complex point-to-point wiring
- 1986: Bosch developed the CAN protocol as a solution
- 1991: Bosch published CAN 2.0 (CAN 2.0A: 11 bit, 2.0B: 29 bit)
- 1993: CAN is adopted as international standard (ISO 11898)
- 2003: ISO 11898 becomes a standard series
- 2012: Bosch released the CAN FD 1.0 (flexible data rate)
- 2015: The CAN FD protocol is standardized (ISO 11898-1)
- 2016: The physical CAN layer for data-rates up to 5 Mbit/s standardized in ISO 11898-2

Today, CAN is standard in automotives (cars, trucks, buses, tractors, ...), ships, planes, EV batteries, machinery and more.



The future of CAN bus

Looking ahead, the CAN bus protocol will stay relevant - though it will be impacted by major trends:

- A need for increasingly advanced vehicle functionality
- The rise of cloud computing
- Growth in Internet of Things (IoT) and connected vehicles
- The impact of autonomous vehicles

In particular, the rise in connected vehicles (V2X) and cloud will lead to a rapid growth in vehicle telematics and IoT CAN loggers. In turn, bringing the CAN bus network 'online' also exposes vehicles to security risks - and may require a shift to new CAN protocols like CAN FD.

The rise of CAN FD

As vehicle functionality expands, so does the load on the CANbus. To support this, CAN FD (Flexible Data Rate) has been designed as the 'next generation' CAN bus. Specifically, CAN FD offers three benefits (vs Classical CAN):

- It enables data rates up to 8 Mbit/s (vs 1 Mbit/s)
- It allows data payloads of up to 64 bytes (vs 8 bytes)
- It enables improved security via authentication

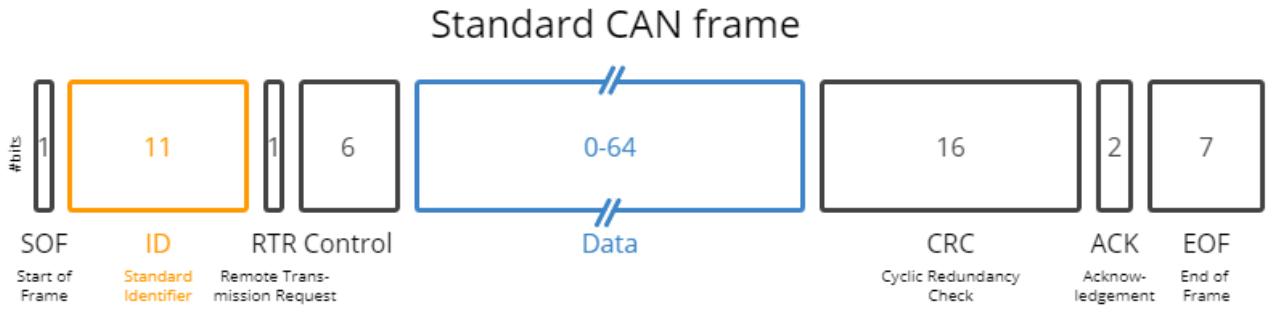
In short, CAN FD boosts speed and efficiency - and it is therefore being rolled out in newer vehicles. This will also drive an increasing need for IoT CAN FD data loggers.

"The first cars using CAN FD will appear in 2019/2020 and CAN FD will replace step-by-step Classical CAN"
- CAN in Automation (CiA), "CAN 2020: The Future of CAN Technology"

What is a CAN frame?

Communication over the CAN bus is done via CAN frames.

Below is a standard CAN frame with 11 bits identifier (CAN 2.0A), which is the type used in most cars. The extended 29-bit identifier frame (CAN 2.0B) is identical except the longer ID. It is e.g. used in the J1939 protocol for heavy-duty vehicles. Note that the CAN ID and Data are highlighted - these are important when recording CAN bus data, as we'll see below.



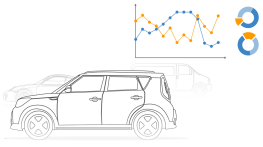
The 8 CAN bus protocol message fields

- **SOF**: The Start of Frame is a 'dominant 0' to tell the other nodes that a CAN node intends to talk
- **ID**: The ID is the frame identifier - lower values have higher priority

- **RTR**: The Remote Transmission Request indicates whether a node sends data or requests dedicated data from another node
- **Control**: The Control contains the Identifier Extension Bit (IDE) which is a 'dominant 0' for 11-bit. It also contains the 4 bit Data Length Code (DLC) that specifies the length of the data bytes to be transmitted (0 to 8 bytes)
- **Data**: The Data contains the data bytes aka payload, which includes CAN signals that can be extracted and decoded for information
- **CRC**: The Cyclic Redundancy Check is used to ensure data integrity
- **ACK**: The ACK slot indicates if the node has acknowledged and received the data correctly
- **EOF**: The EOF marks the end of the CAN frame

Logging CAN data - example use cases

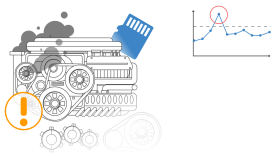
There are several common use cases for recording CAN bus data frames:



Logging/streaming data from cars
OBD2 data from cars can e.g. be used to reduce fuel costs, improve driving, test prototype parts and insurance
learn more

Heavy duty fleet telematics
J1939 data from trucks, buses, tractors etc. can be used in fleet management to reduce costs or improve safety
learn more

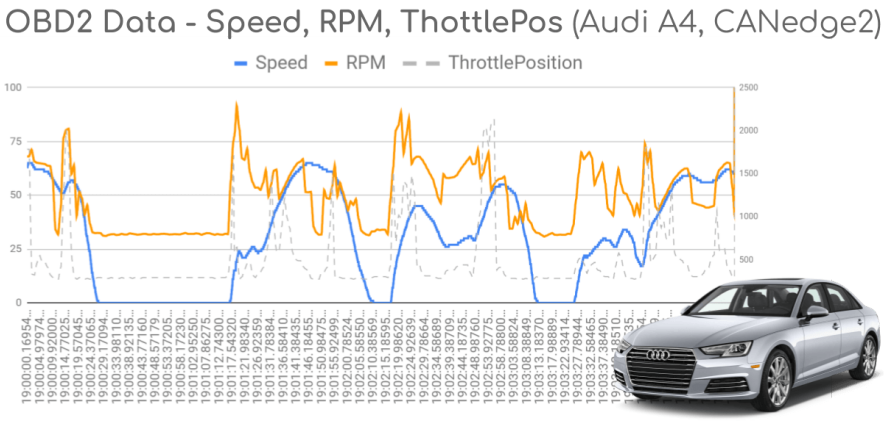
Predictive maintenance
Vehicles and machinery can be monitored via IoT CAN loggers in the cloud to predict and avoid breakdowns
learn more



Vehicle/machine blackbox
A CAN logger can serve as a 'blackbox' for vehicles or equipment, providing data for e.g. disputes or diagnostics
learn more

How to log CAN bus data

As mentioned, two CAN fields are important for CAN logging: The CAN ID and the Data. To record CAN data you need a CAN logger. This lets you log timestamped CAN data to an SD card. In some cases, you need a CAN interface to stream data to a PC - e.g. for car hacking.



Connecting to the CAN bus

This first step is to connect your CAN logger to your CAN bus. Typically this involves using an adapter cable:

- **Cars**: In most cars, you simply use an OBD2 adapter to connect. In most cars, this will let you log raw CAN data, as well as perform requests to log OBD2 or UDS (Unified Diagnostic Services) data
- **Heavy duty vehicles**: To log J1939 data from trucks, excavators, tractors etc you can typically connect to the J1939 CAN bus via a standard J1939 connector cable (deutsch 9-pin)
- **Maritime**: Most ships/boats use the NMEA 2000 protocol and enable connection via an M12 adapter to log marine data
- **CANopen**: For CANopen logging, you can often directly use the CiA 303-1 DB9 connector (i.e. the default connector for our CAN loggers), optionally with a CAN bus extension cable
- **Contactless**: If no connector is available, a typical solution is to use a contactless CAN reader - e.g. the CANCrocodile. This lets you log data directly from the raw CAN twisted wiring harness, without direct connection to the CAN bus (often useful for warranty purposes)
- **Other**: In practice, countless other connectors are used and often you'll need to create a custom CAN bus adapter - here a generic open-wire adapter is useful

When you've identified the right connector and verified the pin-out, you can connect your CAN logger to start recording data. For the CANedge/CLX000, the CAN baud rate is auto-detected and the device will start logging raw CAN data immediately.

Example: Raw CAN sample data (J1939)

You can optionally download raw OBD2 and J1939 samples from the CANedge2 in our intro docs. You can e.g. load this data in the free CAN bus decoder software tools. Data from the CANedge is recorded in the popular binary format, MF4, but can be converted to any file format via our simple MF4 converters (e.g. to CSV, ASC, TRC, ...).

Below is a CSV example of raw CAN frames logged from a heavy-duty truck using the J1939 protocol. Notice that the CAN IDs and data bytes are in hexadecimal format:

```
TimestampEpoch;BusChannel;ID;IDE;DLC;DataLength;Dir;EDL;BRS;DataBytes
1578922367.777150;1;14FEF131;1;8;8;0;0;0;CFFFFFF300FFFF30
1578922367.777750;1;10F01A01;1;8;8;0;0;0;2448FFFFFFFFFFF
1578922367.778300;1;CF00400;1;8;8;0;0;0;107D82BD1200F482
1578922367.778900;1;14FF0121;1;8;8;0;0;0;FFFFFFFFFFFFFCFF
...
```

Example: CANedge CAN logger

The CANedge1 lets you easily record data from any CAN bus to an 8-32 GB SD card. Simply connect it to e.g. a car or truck to start logging - and decode the data via free software/APIs. Further, the CANedge2 (WiFi) and CANedge3 (3G/4G) let you auto-transfer data to your own server - and update devices over-the-air.

How to decode raw CAN data to 'physical values'

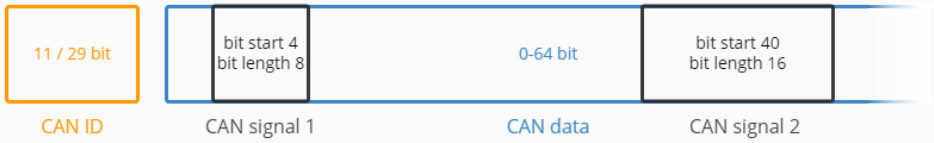
If you review the raw CAN bus data sample above, you will probably notice something: Raw CAN bus data is not human-readable.

To interpret it, you need to decode the CAN frames into scaled engineering values aka physical values (km/h, degC, ...).

Below we show step-by-step how this works.

Extracting CAN signals from raw CAN frames

Each CAN frame on the bus contains a number of CAN signals (parameters) within the CAN databytes. For example, a CAN frame with a specific CAN ID may carry data for e.g. 2 CAN signals.



To extract the physical value of a CAN signal, the following information is required:

- Bit start: Which bit the signal starts at
- Bit length: The length of the signal in bits
- Offset: Value to offset the signal value by
- Scale: Value to multiply the signal value by

To extract a CAN signal, you 'carve out' the relevant bits, take the decimal value and perform a linear scaling:
physical_value = offset + scale * raw_value_decimal

The challenge of proprietary CAN data

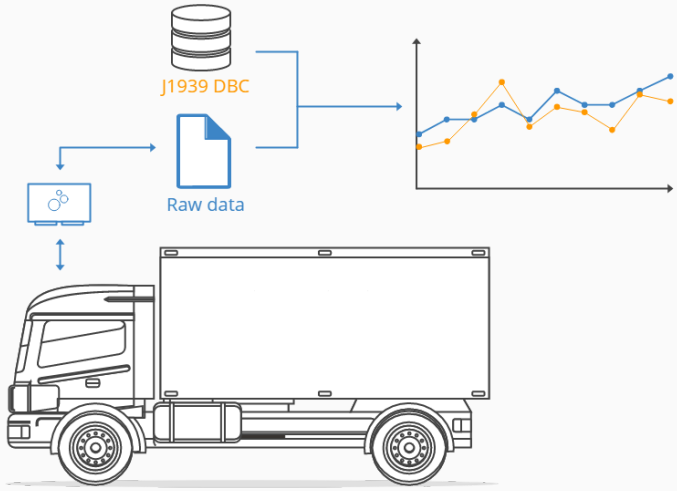
Most often, the CAN bus "decoding rules" are proprietary and not easily available (except to the OEM, i.e. Original Equipment Manufacturer). There are a number of solutions to this when you're not the OEM:

- **Record J1939 data**: If you're logging raw data from heavy duty vehicles (trucks, tractors, ...), you're typically recording J1939 data. This is standardized across brands - and you can use our J1939 DBC file to decode data. See also our J1939 data logger intro
- **Record OBD2/UDS data**: If you need to log data from cars, you can typically request OBD2/UDS data, which is a standardized protocol across cars. For details see our OBD2 data logger intro and our free OBD2 DBC file
- **Use public DBC files**: For cars, online databases exist where others have reverse engineered proprietary some of the CAN data. We keep a list of such databases in our DBC file intro
- **Reverse engineer data**: You can also attempt to reverse engineer data yourself by using a CAN bus sniffer, though it can be time consuming and difficult

- **Use sensor modules**: In some cases you may need sensor data that is not available on the CAN bus (or which is difficult to reverse engineer). Here, sensor-to-CAN modules like the CANmod series can be used. You can integrate such modules with your CAN bus, or use them as add-ons for your CAN logger to add data such as GNSS/IMU or temperature data
- **Partner with the OEM**: In some cases the OEM will provide decoding rules as part of the CAN bus system technical specs. In other cases you may be able to get the information through e.g. a partnership

CAN database files (DBC) - J1939 example

In some cases, conversion rules are standard across manufacturers - e.g. in the J1939 protocol for heavy-duty.
This means that you can use the J1939 parameter conversion rules on practically any heavy-duty vehicle to convert a large share of your data. To make this practical, you need a format for storing the conversion rules. Here, the CAN database (DBC) format is the industry standard - and is supported by most CAN bus decoder software incl. the supporting tools for our CAN loggers, asammdf and CANvas. We also offer a low cost J1939 DBC file, which you can purchase as a digital download. With this, you can get quickly from raw J1939 data to human-readable form. Learn more!



Example: Decoded CAN sample data (physical values)

To illustrate how you can extract CAN signals from raw CAN data frames, we include below the previous J1939 sample data - but now decoded via a J1939 DBC file using the asammdf GUI tool.
As evident, the result is timeseries data with parameters like oil temperature, engine speed, GPS, fuel rate and speed:

```
timestamps,ActualEnginePercentTorque,EngineSpeed,EngineCoolantTemperature,EngineOilTemperature1,EngineFu
elRate,EngineTotalIdleHours,FuelLevel1,Latitude,Longitude,WheelBasedVehicleSpeed
2020-01-13 16:00:13.259449959+01:00,0,1520.13,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
2020-01-13 16:00:13.268850088+01:00,0,1522.88,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
2020-01-13 16:00:13.270649910+01:00,0,1523.34,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
2020-01-13 16:00:13.271549940+01:00,0,1523.58,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
2020-01-13 16:00:13.278949976+01:00,0,1525.5,92,106,3.8,1868.3,52,40.6440124,-76.1223603,86.23
…
```

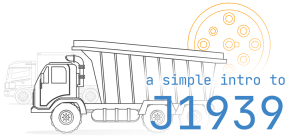
For more on logging J1939 data, see our J1939 data logger and mining telematics articles. You can also learn how to analyze/visualize your CAN data via the free asammdf GUI tool or telematics dashboards.

What is the link between CAN, J1939, OBD2, CANopen ...?

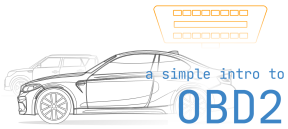
The Controller Area Network provides the basis for communication - but not a lot more.

For example, the CAN standard does not specify how to handle messages larger than 8 bytes - or how to decode the raw data. Therefore a set of **standardized protocols** exist to further specify how data is communicated between CAN nodes of a given network.

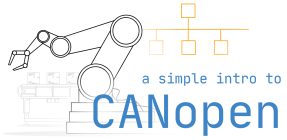
Some of the most common standards include SAE J1939, OBD2 and CANopen. Further, these higher-layer protocols will increasingly be based on the 'next generation' of CAN, CAN FD (e.g. CANopen FD and J1939-17/22).



SAE J1939
J1939 is the standard in-vehicle network for heavy-duty vehicles (e.g. trucks & buses). J1939 parameters (e.g. RPM, speed, ...) are identified by a suspect parameter number (SPN), which are grouped in parameter groups identified by a PG number (PGN).
j1939 intro
j1939 telematics

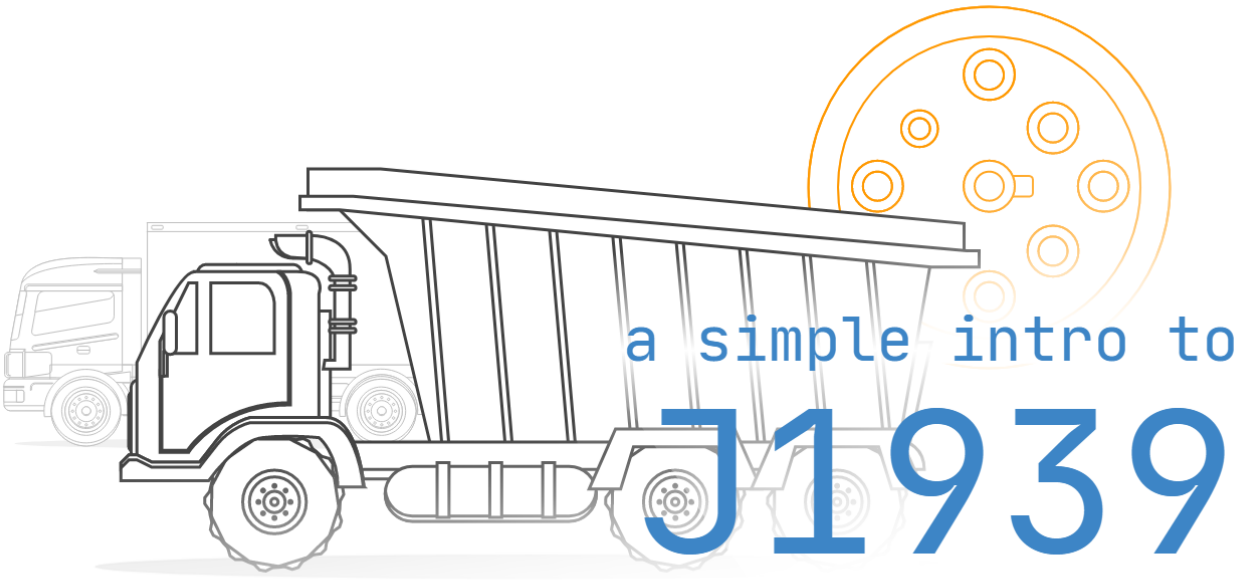


OBD2
On-board diagnostics (OBD, ISO 15765) is a self-diagnostic and reporting capability that e.g. mechanics use to identify car issues. OBD2 specifies diagnostic trouble codes (DTCs) and real-time data (e.g. speed, RPM), which can be recorded via OBD2 loggers.
obd2 intro
obd2 logging



CANopen
CANopen is used widely in embedded control applications, incl. e.g. industrial automation. It is based on CAN, meaning that a CAN bus data logger is also able to log CANopen data. This is key in e.g. machine diagnostics or optimizing production.
canopen intro
canopen logger

CAN FD
CAN bus with flexible data-rate (CAN FD) is an extension of the Classical CAN data link layer. It increases the payload from 8 to 64 bytes and allows for a higher data bit rate, dependent on the CAN transceiver. This enables increasingly data-intensive use cases like EVs.
can fd intro
can fd logger



J1939 Explained - A Simple Intro

In this guide we introduce the J1939 protocol basics incl. PGNs and SPNs. This is a practical intro so you will also learn how to decode J1939 data via DBC files, how J1939 logging works, key use cases and practical tips.

What is J1939?

In short, SAE J1939 is a set of standards that define how ECUs communicate via the CAN bus in heavy-duty vehicles.
As explained in our CAN bus intro, most vehicles today use the Controller Area Network (CAN) for ECU communication.
However, CAN bus only provides a "basis" for communication (like a telephone) - not a "language" for conversation.

In most heavy-duty vehicles, this language is the SAE J1939 standard defined by the Society of Automotive Engineers (SAE). In more technical terms, J1939 provides a higher layer protocol (HLP) based on CAN as the "physical layer".
What does that mean, though?

One standard across heavy-duty vehicles

In simple terms, J1939 offers a standardized method for communication across ECUs, or in other words:
J1939 provides a common language across manufacturers.
In contrast, e.g. cars use proprietary OEM specific protocols.

J1939 application examples

Heavy-duty vehicles (e.g. trucks and buses) is one of the most well-known applications. However, several other key industries leverage SAE J1939 today either directly or via derived standards (e.g. ISO 11783, MilCAN, NMEA 2000, FMS):

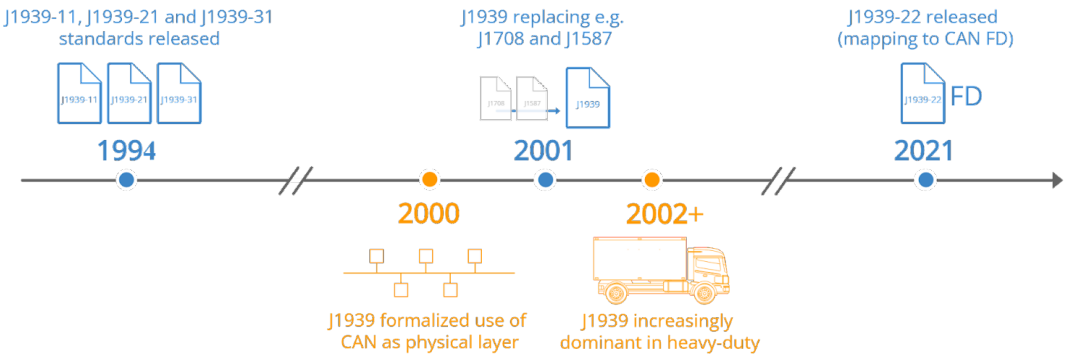
- Foresting machines (e.g. delimbers, forwarders, skidders)
- Mining vehicles (e.g. bulldozers, draglines, excavators, ...)
- Military vehicles (e.g. tanks, transport vehicles, ...)
- Agriculture (e.g. tractors, harvesters, ...)

- Construction (e.g. mobile hydraulics, cranes, ...)
- Fire & Rescue (e.g. ambulances, fire trucks, ...)
- Other (e.g. ships, pumping, e-buses, power generation, ...)

J1939 history & future

History

- 1994: First docs were released (J1939-11, J1939-21, J1939-31)
- 2000: The initial top level document was published
- 2000: CAN formally included as part of J1939 standard
- 2001: J1939 starts replacing former standards SAE J1708/J1587



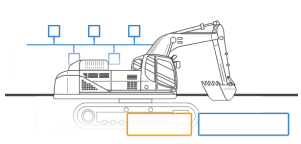
Future

With the rise of heavy-duty telematics, J1939 will increasingly play a role in the market for connected vehicles. In turn, this will increase the need for secure J1939 IoT loggers. In parallel, OEMs will increasingly shift from Classical CAN to CAN FD as part of the transition to J1939 with flexible data-rate. In turn, this will increase the need for J1939 FD data loggers.

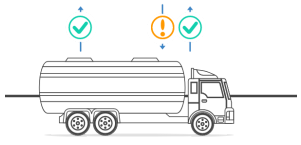
"The market for in-vehicle connectivity - the hardware and services bringing all kinds of new functionality to drivers and fleet owners - is expected to reach EUR 120 billion by 2020."
- Boston Consulting Group, Connected Vehicles and the Road to Revenue

4 key characteristics of J1939

The J1939 protocol has a set of defining characteristics outlined below:

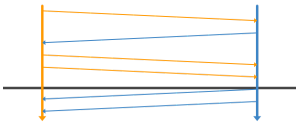


250K baud rate & 29-bit extended ID
The J1939 baud rate is typically 250K (though recently with support for 500K) - and the identifier is extended 29-bit (CAN 2.0B)



Broadcast + on-request data
Most J1939 messages are broadcast on the CAN-bus, though some data is only available by requesting the data via the CAN bus

PGN identifiers & SPN parameters
J1939 messages are identified by 18-bit Parameter Group Numbers (PGN), while J1939 signals are called Suspect Parameter Numbers (SPN)



Multibyte variables & Multi-packets
Multibyte variables are sent least significant byte first (Intel byte order). PGNs with up to 1785 bytes are supported via J1939 transport protocol

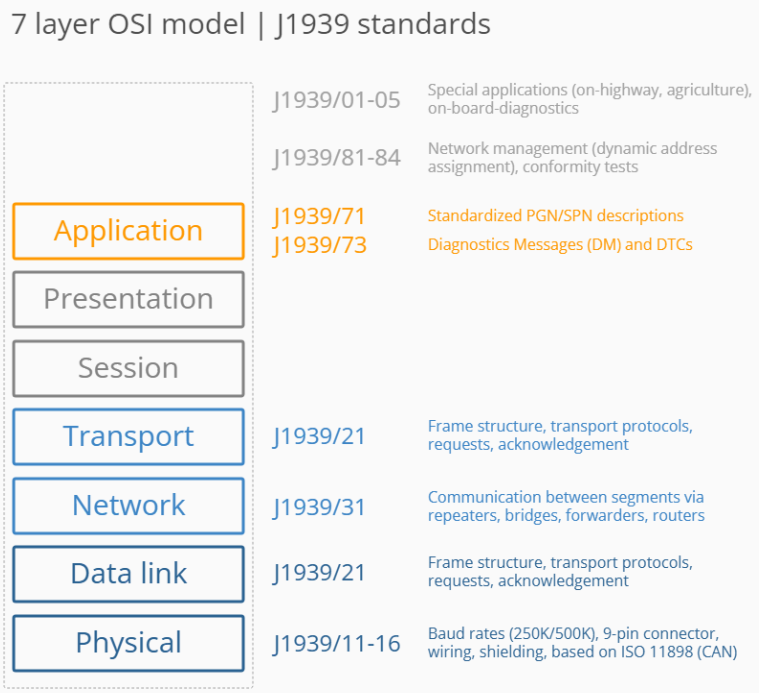
Additional J1939 characteristics

Below are a set of additional characteristics of the J1939 protocol:

- Reserved: J1939 includes a large range of standard PGNs, though PGNs 00FF00 through 00FFFF are reserved for proprietary use
- Special Values: A data byte of 0xFF (255) reflects N/A data, while 0xFE (254) reflects an error
- J1939 address claim: The SAE J1939 standard defines a procedure for assigning source addresses to J1939 ECUs after network initialization via an 8-bit address in a dynamic way

Technical: J1939 'higher layer protocol' explained

J1939 is based on CAN, which provides the basic "physical layer" and "data link layer", the lowest layers in the OSI model. Basically, CAN allows the communication of small packets on the CAN bus, but not a lot more than that. Here, J1939 serves as a higher layer protocol on top, enabling more complex communication.



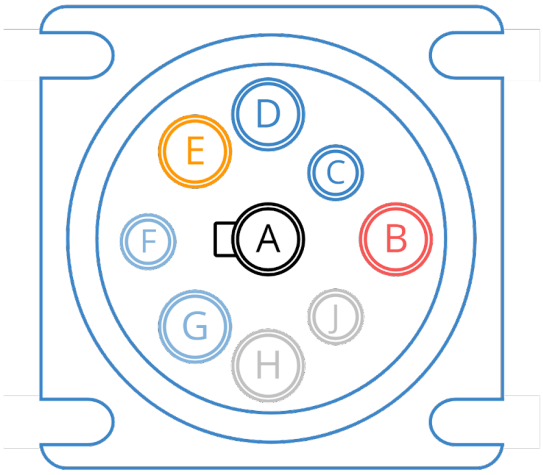
A higher layer protocol enables communication across the large complex networks of e.g. vehicle manufacturers.

For example, the SAE J1939 protocol specifies how to handle "multi-packet messages", i.e. when data larger than 8 bytes needs to be transferred. Similarly, it specifies how data is to be converted into human-readable data.
It does so by providing a family of standards. For example, J1939-71 is a document detailing the information required to convert a large set of cross-manufacturer standardized J1939 messages into human-readable data (more on this below). Many other CAN based higher layer protocols exist, e.g. CANopen, DeviceNet, Unified Diagnostic Services. These typically offer some level of standardization within their respective industries - though all of them can be extended by manufacturers.

In comparison, the aforementioned passenger cars have unique standards per manufacturer. In other words, you can use the same J1939 database file to convert e.g. engine speed across two trucks from different manufacturers - but you cannot e.g. read the data from an Audi A4 using the same IDs & scaling parameters as for a Peugeot 207.

The J1939 connector (9-pin)

The J1939-13 standard specifies the 'off-board diagnostic connector' - also known as the J1939 connector or 9-pin deutsch connector. This is a standardized method for interfacing with the J1939 network of most heavy duty vehicles - see the illustration for the J1939 connector pinout.



A	Ground
B	Battery power
C	CAN 1 H
D	CAN 1 L
E	CAN shield
F	J1708 (+) / CAN 2 H
G	J1708 (-) / CAN 2 L
H	OEM specific
J	OEM specific

Type 1 (black): CAN 1 = 250K
Type 2 (green): CAN 1 = 500K

Black type 1 vs green type 2

Note that the J1939 deutsch connector comes in two variants: The original black connector (type 1) and the newer green connector (type 2), which started getting rolled out around 2013-14.

J1939 type 2 female connectors are physically backwards compatible, while type 1 female connectors only fit type 1 male sockets. The type 2 connector was designed for the SAE J1939-14 standard, which adds support for 500K bit rates. The purpose of "blocking" type 1 connectors is to ensure that older hardware (presumably using 250K bit rates) is not connected to type 2 500K bit rate J1939 networks. Specifically, the physical block is through a smaller hole for pin F in the type 2 male connectors. See also the example of a DB9-J1939 adapter cable (type 2).

Multiple J1939 networks

As evident, the J1939 deutsch connector provides access to the J1939 network through pins C (CAN high) and D (CAN low). This makes it easy to interface with the J1939 network across most heavy duty vehicles.

In some cases, however, you may also be able to access a secondary J1939 network through pins F and G or pins H and J (with H being CAN H and J being CAN L).

Many of today's heavy duty vehicles have 2 or more parallel CAN bus networks and in some cases at least two of these will be available through the same J1939 connector. This also means that you will not necessarily have gained access to all the available J1939 data if you've only attempted to interface through the 'standard' pins C and D.

Other heavy duty connectors

While the J1939 deutsch connector is the most common way to interface with the J1939 network of heavy duty vehicles, other connectors of course exist. Below are some examples:

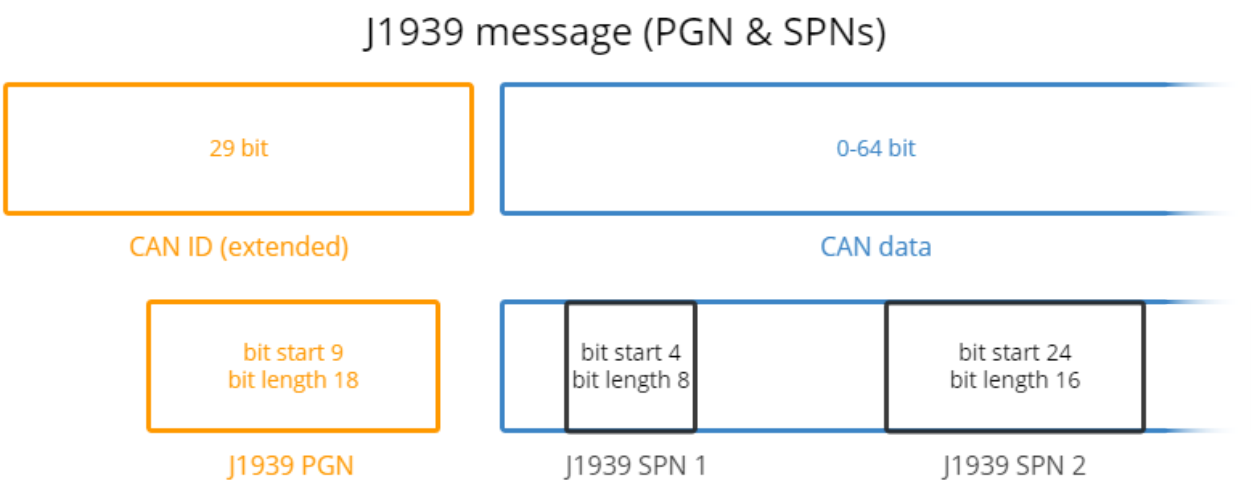
- J1939 Backbone Connector: This 3-pin deutsch connector provides pins for CAN H/L a CAN shield (no power/ground)
- CAT connector: The Caterpillar industrial connector is a grey 9-pin deutsch connector. However, the pin-out differs from the J1939 connector (A: Power, B: Ground, F: CAN L, G: CAN H) and the connector physically blocks access from standard type 1 and 2 J1939 connectors
- OBD2 type B connector: The type B OBD2 connector (SAE J1962) in heavy duty vehicles sometimes provide direct access to the J1939 network through pins 6 and 14
- Volvo 2013 OBD2 connector: This variant matches the type B OBD2 connector, but also adds access to the J1939 high via pin 3 and J1939 low via pin 11

The J1939 PGN and SPN

In the following section we explain the J1939 PGNs and SPNs.

Parameter Group Number (PGN)

The J1939 PGN comprises an 18-bit subset of the 29-bit extended CAN ID. In simple terms, the PGN serves as a unique frame identifier within the J1939 standard. For example, you can look this up in the J1939-71 standard documentation, which lists PGNs/SPNs.



Example: J1939 PGN 61444 (EEC1)

Assume you recorded a J1939 message with HEX ID 0CF00401. Here, the PGN starts at bit 9, with length 18 (indexed from 1). The resulting PGN is 0F004 or in decimal 61444. Looking this up in the SAE J1939-71 documentation, you will find that it is the "Electronic Engine Controller 1 - EEC1". Further, the document will have details on the PGN including priority, transmission rate and a list of the associated SPNs - cf. the illustration. For this PGN, there are seven SPNs (e.g. Engine Speed, RPM), each of which can be looked up in the J1939-71 documentation for further details.

PGN61444 - Electronic Engine Controller 1 - EEC1

Transmission Repetition Rate: Engine speed dependent
Data Length: 8 bytes
Data Page: 0
PDU Format: 240
PDU Specific: 4
Default Priority: 3
Parameter Group Number: 61444 (0x00F004)

Bit Start/Byte	Length	SPN ID	SPN Description
1.1	4 bits	899	Engine Torque Mode
2	1 byte	512	Driver's Demand Engine - % Torque
3	1 byte	513	Actual Engine - Percent Torque
4-5	2 bytes	190	Engine Speed
6	1 byte	1483	SA of Controlling Device for Engine Control
7.1	4 bits	1675	Engine Starter Mode
8	1 byte	2432	Engine Demand - Percent Torque

Detailed breakdown of the J1939 PGN

Let's look at the CAN ID to PGN transition in detail. Specifically, the 29 bit CAN ID comprises the Priority (3 bits), the J1939 PGN (18 bits) and the Source Address (8 bits). In turn, the PGN can be split into the Reserved Bit (1 bit), Data Page (1 bit), PDU format (8 bit) and PDU Specific (8 bit).

The detailed PGN illustration also includes example values for each field in binary, decimal and hexadecimal form.

To learn more about the transition from 29 bit CAN ID to 18 bit J1939 PGN, see also our online CAN ID to J1939 PGN converter. The converter also includes a full J1939 PGN list for PGNs included in our J1939 DBC file.