Конец формы

From Wikipedia, the free encyclopedia

**LIN** (**Local Interconnect Network**) is a [network protocol](https://en.wikipedia.org/wiki/Network_protocol) used for communication between components in modern [vehicles](https://en.wikipedia.org/wiki/Vehicles). It is a low-cost single-wire [serial protocol](https://en.wikipedia.org/wiki/Serial_communication) that supports communications up to 19.2 Kbit/s with a maximum bus length of 40 metres (131.2 ft).

**History**

[[edit](https://en.wikipedia.org/w/index.php?title=Local_Interconnect_Network&action=edit&section=1)]

The need for a cheap serial network arose as the technologies and the facilities implemented in the car grew, while the [CAN bus](https://en.wikipedia.org/wiki/CAN_bus) was too expensive to implement for every component in the car. European car manufacturers started using different serial communication technologies, which led to compatibility problems.

In the late 1990s, the LIN Consortium was founded by five automakers ([BMW](https://en.wikipedia.org/wiki/BMW), [Volkswagen Group](https://en.wikipedia.org/wiki/Volkswagen_Group), [Audi](https://en.wikipedia.org/wiki/Audi), [Volvo Cars](https://en.wikipedia.org/wiki/Volvo_Cars), [Mercedes-Benz](https://en.wikipedia.org/wiki/Mercedes-Benz)), with the technologies supplied (networking and hardware expertise) from Volcano Automotive Group and [Motorola](https://en.wikipedia.org/wiki/Motorola). The first fully implemented version of the new LIN specification (LIN version 1.3) was published in November 2002. In September 2003, version 2.0 was introduced to expand capabilities and make provisions for additional diagnostics features. LIN may be used also over the vehicle's battery [power line](https://en.wikipedia.org/wiki/Power_line_communication#Automotive_uses) with a special LIN-over-DC-power-line (DC-LIN) transceiver. LIN over DC power line (DC-LIN) was standardized as ISO/AWI 17987-8.[[1]](https://en.wikipedia.org/wiki/Local_Interconnect_Network#cite_note-1)

[CAN in Automation](https://en.wikipedia.org/wiki/CAN_in_Automation) has been appointed by the ISO Technical Management Board (TMB) as the Registration Authority for the LIN Supplier ID standardized in the ISO 17987 series.

**Network topology**

[[edit](https://en.wikipedia.org/w/index.php?title=Local_Interconnect_Network&action=edit&section=2)]

LIN is a [broadcast](https://en.wikipedia.org/wiki/Broadcasting_(networking)) [serial](https://en.wikipedia.org/wiki/Serial_communications) network comprising 16 nodes (one master and up to 15 slaves).[[2]](https://en.wikipedia.org/wiki/Local_Interconnect_Network#cite_note-2)[[3]](https://en.wikipedia.org/wiki/Local_Interconnect_Network#cite_note-3)[[4]](https://en.wikipedia.org/wiki/Local_Interconnect_Network#cite_note-4)[[5]](https://en.wikipedia.org/wiki/Local_Interconnect_Network#cite_note-5)

All messages are initiated by the master with at most one slave replying to a given message identifier. The master node can also act as a slave by replying to its own messages. Because all communications are initiated by the master it is not necessary to implement a [collision](https://en.wikipedia.org/wiki/Collision_(telecommunications)) detection.[[6]](https://en.wikipedia.org/wiki/Local_Interconnect_Network#cite_note-6)

The master and slaves are typically [microcontrollers](https://en.wikipedia.org/wiki/Microcontroller), but may be implemented in specialized hardware or [ASICs](https://en.wikipedia.org/wiki/Application-specific_integrated_circuit) in order to save cost, space, or power.

Current uses combine the low-cost efficiency of LIN and simple sensors to create small networks. These sub-systems can be connected by a back-bone network (i.e. CAN in cars).[[7]](https://en.wikipedia.org/wiki/Local_Interconnect_Network#cite_note-7)

**Overview**

[[edit](https://en.wikipedia.org/w/index.php?title=Local_Interconnect_Network&action=edit&section=3)]

The LIN bus is an inexpensive serial communications protocol, which effectively supports remote application within a car's network. It is particularly intended for mechatronic nodes in distributed automotive applications, but is equally suited to industrial applications. It is intended to complement the existing CAN network leading to hierarchical networks within cars.

In the late 1990s the Local Interconnect Network (LIN) Consortium was founded by five European automakers, [Mentor Graphics](https://en.wikipedia.org/wiki/Mentor_Graphics) (Formerly Volcano Automotive Group) and [Freescale](https://en.wikipedia.org/wiki/Freescale) (Formerly [Motorola](https://en.wikipedia.org/wiki/Motorola), now [NXP](https://en.wikipedia.org/wiki/NXP)). The first fully implemented version of the new LIN specification was published in November 2002 as LIN version 1.3. In September 2003 version 2.0 was introduced to expand configuration capabilities and make provisions for significant additional diagnostics features and tool interfaces.

The protocol’s main features are listed below:

* Single master, up to 16 slaves (i.e. no bus arbitration). This is the value recommended by the LIN Consortium to achieve deterministic time response.[[8]](https://en.wikipedia.org/wiki/Local_Interconnect_Network#cite_note-clemson_autobuses-8)
  + Slave Node Position Detection (SNPD) allows node address assignment after power-up[[9]](https://en.wikipedia.org/wiki/Local_Interconnect_Network#cite_note-SPEC-9)
* Single-wire communications up to 19.2 kbit/s @ 40 [meter](https://en.wikipedia.org/wiki/Meter) bus length.[[8]](https://en.wikipedia.org/wiki/Local_Interconnect_Network#cite_note-clemson_autobuses-8)[[10]](https://en.wikipedia.org/wiki/Local_Interconnect_Network#cite_note-interfacebus_com-Design_Connector_LIN_Bus-10) In the LIN specification 2.2,[[9]](https://en.wikipedia.org/wiki/Local_Interconnect_Network#cite_note-SPEC-9) the speed up to 20 kbit/s.
* Guaranteed latency times.
* Variable length of data frame (2, 4 and 8 bytes).
* Configuration flexibility.
* Multicast reception with time synchronization, without crystals or [ceramic resonators](https://en.wikipedia.org/wiki/Ceramic_resonator).
* Data checksum and error detection.
* Detection of defective nodes.
* Low-cost silicon implementation based on standard [UART](https://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter)/[SCI](https://en.wikipedia.org/wiki/Serial_communication) hardware.
* Enabler for hierarchical networks.
* Operating voltage of 12 V.[[8]](https://en.wikipedia.org/wiki/Local_Interconnect_Network#cite_note-clemson_autobuses-8)

Data is transferred across the bus in fixed-form messages of selectable lengths. The master task transmits a header that consists of a break signal followed by synchronization and identifier fields. The slaves respond with a data frame that consists of 2, 4 or 8 data bytes plus 3 bytes of control information.[[9]](https://en.wikipedia.org/wiki/Local_Interconnect_Network#cite_note-SPEC-9)

**LIN message frame**

[[edit](https://en.wikipedia.org/w/index.php?title=Local_Interconnect_Network&action=edit&section=4)]

A message contains the following fields:[[9]](https://en.wikipedia.org/wiki/Local_Interconnect_Network#cite_note-SPEC-9)

* Synchronization break
* Synchronization byte
* Identifier byte
* Data bytes
* Checksum byte

**Frame types**

[[edit](https://en.wikipedia.org/w/index.php?title=Local_Interconnect_Network&action=edit&section=5)]

1. **Unconditional frame.**These always carry signals and their identifiers are in the range 0 to 59 (0x00 to 0x3b). All subscribers of the unconditional frame shall receive the frame and make it available to the application (assuming no errors were detected).
2. **Event-triggered frame.** The purpose of this is to increase the responsiveness of the LIN cluster without assigning too much of the bus bandwidth to the polling of multiple slave nodes with seldom occurring events. The first data byte of the carried unconditional frame shall be equal to a protected identifier assigned to an event-triggered frame. A slave shall reply with an associated unconditional frame only if its data value has changed. If none of the slave tasks responds to the header the rest of the frame slot is silent and the header is ignored. If more than one slave task responds to the header in the same frame slot a collision will occur, and the master has to resolve the collision by requesting all associated unconditional frames before requesting the event-triggered frame again.
3. **Sporadic frame.** This frame is transmitted by the master as required, so a collision cannot occur. The header of a sporadic frame shall only be sent in its associated frame slot when the master task knows that a signal carried in the frame has been updated. The publisher of the sporadic frame shall always provide the response to the header.
4. **Diagnostic frame.** These always carry diagnostic or configuration data and they always contain eight data bytes. The identifier is either 60 (0x3C), called master request frame, or 61(0x3D), called slave response frame. Before generating the header of a diagnostic frame, the master task asks its diagnostic module if it shall be sent or if the bus shall be silent. The slave tasks publish and subscribe to the response according to their diagnostic module.
5. **User-defined frame.** These can carry any kind of information. Their identifier is 62 (0x3E). The header of a user-defined frame is always transmitted when a frame slot allocated to the frame is processed
6. **Reserved frame.** These shall not be used in a LIN 2.0 cluster. Their identifier is 63 (0x3F).

**LIN hardware**

[[edit](https://en.wikipedia.org/w/index.php?title=Local_Interconnect_Network&action=edit&section=6)]

The LIN specification was designed to allow very cheap hardware-nodes being used within a network. It is a low-cost, single-wire network based on [ISO 9141](https://en.wikipedia.org/wiki/On-board_diagnostics).[[11]](https://en.wikipedia.org/wiki/Local_Interconnect_Network#cite_note-11) In today’s car networking topologies, microcontrollers with either [UART](https://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter) capability or dedicated LIN hardware are used. The microcontroller generates all needed LIN data (protocol ...) (partly) by software and is connected to the LIN network via a LIN [transceiver](https://en.wikipedia.org/wiki/Transceiver) (simply speaking, a level shifter with some add-ons). Working as a LIN node is only part of the possible functionality. The LIN hardware may include this transceiver and works as a pure LIN node without added functionality.

As LIN Slave nodes should be as cheap as possible, they may generate their internal clocks by using [RC oscillators](https://en.wikipedia.org/wiki/RC_oscillator) instead of [crystal oscillators](https://en.wikipedia.org/wiki/Crystal_oscillator) (quartz or a ceramic). To ensure the [baud](https://en.wikipedia.org/wiki/Baud) rate-stability within one LIN frame, the SYNC field within the header is used.

**LIN protocol**

[[edit](https://en.wikipedia.org/w/index.php?title=Local_Interconnect_Network&action=edit&section=7)]

The LIN-Master uses one or more predefined [scheduling](https://en.wikipedia.org/wiki/I/O_scheduling) tables to start the sending and receiving to the LIN bus. These scheduling tables contain at least the relative timing, where the message sending is initiated. One LIN Frame consists of the two parts **header** and **response**. The header is always sent by the LIN Master, while the response is sent by either one dedicated LIN-Slave or the LIN master itself.

Transmitted data within the LIN is transmitted serially as eight bit data bytes with one start bit, one stop-bit, and no parity (break field does not have a start or stop bit). Bit rates vary within the range of 1 [kbit/s](https://en.wikipedia.org/wiki/Kbit/s) to 20 kbit/s. Data on the bus is divided into recessive (logical HIGH) and dominant (logical LOW). The time normally is considered by the LIN Masters stable clock source, the smallest entity is one [bit time](https://en.wikipedia.org/wiki/Bit_time) (52 μs @ 19.2 kbit/s).

Two bus states – sleep-mode and active – are used within the LIN protocol. While data is on the bus, all LIN-nodes are asked to be in the active state. After a specified timeout, the nodes enter sleep mode and will be released back to active state by a WAKEUP frame. This frame may be sent by any node requesting activity on the bus, either the LIN Master following its internal schedule, or one of the attached LIN Slaves being activated by its internal software application. After all nodes are awakened, the Master continues to schedule the next Identifier.

**Header**

[[edit](https://en.wikipedia.org/w/index.php?title=Local_Interconnect_Network&action=edit&section=8)]

The header consists of five parts:

*BREAK:* The BREAK field is used to activate all attached LIN slaves to listen to the following parts of the header. It consists of one start bit and several dominant bits. The length is at least 11-bit times; standard use as of today are 13-bit times, and therefore differs from the basic data format. This is used to ensure that listening LIN nodes with a main-clock differing from the set bus baud rate in specified ranges will detect the BREAK as the frame starting the communication and not as a standard data byte with all values zero ([hexadecimal](https://en.wikipedia.org/wiki/Hexadecimal) 0x00).

*SYNC:* The SYNC is a standard data format byte with a value of hexadecimal 0x55. LIN slaves running on RC oscillator will use the distance between a fixed amount of rising and falling edges to measure the current bit time on the bus (the master's time normal) and to recalculate the internal baud rate.

*INTER BYTE SPACE:* Inter Byte Space is used to adjust for bus jitter. It is an optional component within the LIN specification. If enabled, then all LIN nodes must be prepared to deal with it.

There is an Inter Byte Space between the BREAK and SYNC field, one between the SYNC and IDENTIFIER, one between the payload and Checksum and one between every Data byte in the payload.

*IDENTIFIER:* The IDENTIFIER defines one action to be fulfilled by one or several of the attached LIN slave nodes. The network designer has to ensure the fault-free functionality in the design phase (one slave is allowed to send data to the bus in one frame time).

If the identifier causes one *physical* LIN slave to send the response, the identifier may be called a Rx-identifier. If the *master's slave task* sends data to the bus, it may be called Tx-identifier.

*RESPONSE SPACE:* Response Space is the time between the IDENTIFIER field and the first Data byte which starts the LIN RESPONSE part of the LIN frame. When a particular LIN frame is transmitted completely, Header + Response, by the LIN MASTER, the LIN MASTER will use the full RESPONSE SPACE TIME to calculate when to send the response after sending the header. If the response part of the LIN frame is coming from a physically different SLAVE NODE, then each node (master & slave) will utilize 50% of the Response Space time in their timeout calculations.

**Response**

[[edit](https://en.wikipedia.org/w/index.php?title=Local_Interconnect_Network&action=edit&section=9)]

The response is sent by one of the attached LIN slave **tasks** and is divided into data and [checksum](https://en.wikipedia.org/wiki/Checksum).[[9]](https://en.wikipedia.org/wiki/Local_Interconnect_Network#cite_note-SPEC-9)

*DATA:* The responding slave may send zero to eight data bytes to the bus. The amount of data is fixed by the application designer and mirrors data relevant for the application which the LIN slave runs in.

*CHECKSUM:* There are two checksum-models available within LIN - The first is the checksum including the data bytes only (specification up to Version 1.3), the second one includes the identifier in addition (Version 2.0+). The used checksum model is pre-defined by the application designer.

**Slave node position detection (SNPD) or autoaddressing**

[[edit](https://en.wikipedia.org/w/index.php?title=Local_Interconnect_Network&action=edit&section=10)]

These methods allow the detection of the position of slave nodes on the LIN bus and allow the assignment of a unique node address (NAD).[[12]](https://en.wikipedia.org/wiki/Local_Interconnect_Network#cite_note-12)

* Allows similar or the same devices to be connected on the bus without end of line programming or connector pin programming.

Restrictions:

* All auto-addressing slaves must be in one line
  + Standard slaves can be connected in any way

|  |  |  |
| --- | --- | --- |
| **SNPD Method** | **SNPD Method ID** | **Company** |
| Extra wire daisy chain | 0x01 | [NXP](https://en.wikipedia.org/wiki/NXP) (formerly Philips) |
| Bus shunt method | 0x02 | [Elmos Semiconductor](https://en.wikipedia.org/wiki/Elmos_Semiconductor) |
| Reserved | 0x03 | TBD |
| Reserved | 0x04 | TBD |
| Reserved | 0xFF | TBD |

**Extra wire daisy chain (XWDC)**

[[edit](https://en.wikipedia.org/w/index.php?title=Local_Interconnect_Network&action=edit&section=11)]

Each slave node has to provide two extra pins, one input, D1, and one output, D2.

* The first SNPD node input D1 is either set to GND or connected to the output of the master.
  + The output of the first node, D2, is connected to the input, D1 of the second node, and so on resulting in a daisy chain.

Each configuration pin Dx (x=1-2) has additional circuitry to aid in the position detection.

1. Switchable resistive pull-up to Vbat
2. Pull-down to GND
3. Comparator referenced to Vbat/2

**XWDC auto-addressing procedure**

[[edit](https://en.wikipedia.org/w/index.php?title=Local_Interconnect_Network&action=edit&section=12)]

At the start of the procedure no SNPD devices have a NAD assigned

1 First auto-addressing LIN message

1.1 All outputs (D2) are set to a high level, all pull-downs are turned off

1.2 The first SNPD node is selected. It is identified by having the input D1 low (hardwired).

1.3 The selected node takes the address from the LIN configuration message

1.4 The detected node turns on the pull-down at the output D2

2 Subsequent auto-addressing LIN messages

2.1 The first non addressed SNPD node is selected. It is identified by having the input D1 low (D2 of previous node).

2.2 The selected node takes the address from the LIN configuration message

2.3 The detected node turns on the pull-down at the output D2

2.4 Steps 2.1-2.4 are repeated until all slave nodes are assigned an address

3 All pull-ups and pull-downs are turned off completing the addressing procedure

**Bus shunt method (BSM)**

[[edit](https://en.wikipedia.org/w/index.php?title=Local_Interconnect_Network&action=edit&section=13)]

Each slave node has two LIN pins

1. bus\_in
2. bus\_out

Each slave node needs some additional circuitry compared to the standard LIN circuitry to aid in the position detection.

1. The standard pull-up must be switchable
2. Switchable 2 mA current source from Vbat
3. Shunt resistor
4. Differential amplifier
5. Analog to digital converter

**BSM auto-addressing procedure**

[[edit](https://en.wikipedia.org/w/index.php?title=Local_Interconnect_Network&action=edit&section=14)]

At the start of the procedure, none of the SNPD devices have a NAD assigned. The autoaddressing routine is performed during the sync field. The sync field is broken into three phases:

1 Offset current measurement

1.1 All outputs pull-ups and current sources are switched off

1.2 The bus current is measured, **Ioffset**

2 Pull-up mode

2.1 Pull-ups are turned on and current sources remain off

2.2 The bus current is measured, **IPU**

2.3 Nodes with ΔI = **IPU**-**Ioffset** < 1 mA are "selected"

3 Current source mode

3.1 Selected nodes switch current source on and others switch pull-ups off

3.2 Bus current is measured, **ICS**

3.3 Node with ΔI = **ICS**-**Ioffset** < 1 mA is detected as the last node

3.4 Current sources are switched off and pull-ups are switched on

3.5 The last node will accept the address contained in the LIN configuration message

This technique is covered by the patents EP 1490772 B1 and US 7091876.

**LIN advantages**

[[edit](https://en.wikipedia.org/w/index.php?title=Local_Interconnect_Network&action=edit&section=15)]

* Easy to use
* Components available
* Cheaper than CAN and other communications buses
* Harness reduction
* More reliable vehicles
* Extension easy to implement.
* No protocol license fee required

LIN is not a full replacement of the CAN bus. But the LIN bus is a good alternative wherever low costs are essential and speed/bandwidth is not important. Typically, it is used within sub-systems that are not critical to vehicle performance or safety - some examples are given below.

**Applications**

[[edit](https://en.wikipedia.org/w/index.php?title=Local_Interconnect_Network&action=edit&section=16)]

|  |  |
| --- | --- |
| **Application segments** | **Specific LIN application examples** |
| Roof | Sensor, light sensor, light control, sun roof |
| Steering wheel | Cruise control, wiper, turning light, climate control, radio, wheel lock |
| Seat | Seat position motors, occupant sensors, control panel |
| Engine | Sensors, small motors, cooling fan motors |
| Grille | Grille shutter |
| Climate | Small motors, control panel |
| Door | Mirror, central ECU, mirror switch, window lift, seat control switch, door lock |
| Illumination | Vehicle trim enhancement, sill plates illuminated with RGB LED |

**Addressing**

[[edit](https://en.wikipedia.org/w/index.php?title=Local_Interconnect_Network&action=edit&section=17)]

Addressing in LIN is achieved with a NAD (Node ADdress) that is part of the PID (protected identifier). NAD values are on 7bits, so in the range 1 to 127 (0x7F) and it is a composition of supplier ID, function ID and variant ID.

You can obtain a supplier ID by contacting [CAN in Automation](https://en.wikipedia.org/wiki/CAN_in_Automation) that is the authority responsible for the assignment of such identifiers.

-------------------------------------------------------------------------------------------------------------------

**FlexRay** is an [automotive network communications protocol](https://en.wikipedia.org/wiki/Vehicle_bus) developed by the **FlexRay Consortium** to govern on-board automotive computing. It is designed to be faster and more reliable than [CAN](https://en.wikipedia.org/wiki/Controller_Area_Network) and [TTP](https://en.wikipedia.org/wiki/Time-Triggered_Protocol), but it is also more expensive. The FlexRay consortium disbanded in 2009, but the FlexRay standard is now a set of ISO standards, ISO 17458-1[[1]](https://en.wikipedia.org/wiki/FlexRay#cite_note-1) to 17458-5.[[2]](https://en.wikipedia.org/wiki/FlexRay#cite_note-2)[[3]](https://en.wikipedia.org/wiki/FlexRay#cite_note-3)

FlexRay is a communication bus designed to ensure high data rates, fault tolerance, operating on a time cycle, split into static and dynamic segments for event-triggered and time-triggered communications. It is mainly used in aeronautic and automotive sectors.

**Features**

[[edit](https://en.wikipedia.org/w/index.php?title=FlexRay&action=edit&section=1)]

FlexRay supports data rates up to 10 [Mbit/s](https://en.wikipedia.org/wiki/Megabit), explicitly supports both star and bus physical topologies, and can have two independent data channels for fault-tolerance (communication can continue with reduced bandwidth if one channel is inoperative). The bus operates on a time cycle, divided into two parts: the static segment and the dynamic segment. The static segment is preallocated into slices for individual communication types, providing stronger determinism than its predecessor [CAN](https://en.wikipedia.org/wiki/Controller_Area_Network). The dynamic segment operates more like CAN, with nodes taking control of the bus as available, allowing event-triggered behavior.[[4]](https://en.wikipedia.org/wiki/FlexRay#cite_note-4)

**Consortium**

[[edit](https://en.wikipedia.org/w/index.php?title=FlexRay&action=edit&section=2)]

The FlexRay Consortium was made up of the following core members:

* [Freescale Semiconductor](https://en.wikipedia.org/wiki/Freescale_Semiconductor)
* [Bosch](https://en.wikipedia.org/wiki/Robert_Bosch_GmbH)
* [NXP Semiconductors](https://en.wikipedia.org/wiki/NXP_Semiconductors)
* [BMW](https://en.wikipedia.org/wiki/BMW)
* [Volkswagen](https://en.wikipedia.org/wiki/Volkswagen_Group)
* [Daimler](https://en.wikipedia.org/wiki/Daimler_AG)
* [General Motors](https://en.wikipedia.org/wiki/General_Motors)

There were also Premium Associate and Associate members of FlexRay consortium. By September 2009, there were 28 premium associate members and more than 60 associate members. At the end of 2009, the consortium disbanded.

**Commercial deployment**

[[edit](https://en.wikipedia.org/w/index.php?title=FlexRay&action=edit&section=3)]

The first series production vehicle with FlexRay was at the end of 2006 in the [BMW X5 (E70)](https://en.wikipedia.org/wiki/BMW_X5_(E70)),[[5]](https://en.wikipedia.org/wiki/FlexRay#cite_note-:0-5) enabling a new and fast adaptive damping system. Full use of FlexRay was introduced in 2008 in the new [BMW 7 Series (F01)](https://en.wikipedia.org/wiki/BMW_7_Series_(F01)).

**Vehicles**

[[edit](https://en.wikipedia.org/w/index.php?title=FlexRay&action=edit&section=4)]

* [Audi A4 (B9)](https://en.wikipedia.org/wiki/Audi_A4) (2015–)[[6]](https://en.wikipedia.org/wiki/FlexRay#cite_note-6)
* [Audi A5 (F5)](https://en.wikipedia.org/wiki/Audi_A5) (2016–)[[7]](https://en.wikipedia.org/wiki/FlexRay#cite_note-7)
* [Audi A6 (C7)](https://en.wikipedia.org/wiki/Audi_A6_C7) (2011–2018)[[8]](https://en.wikipedia.org/wiki/FlexRay#cite_note-8)
* [Audi A7](https://en.wikipedia.org/wiki/Audi_A7)
* [Audi A8 (D4)](https://en.wikipedia.org/wiki/Audi_A8_D4) (2010–2017)[[9]](https://en.wikipedia.org/wiki/FlexRay#cite_note-9)
* [Audi Q7](https://en.wikipedia.org/wiki/Audi_Q7#Second_generation_(Typ_4M;_2015)) (2015–)
* [Audi TT Mk3](https://en.wikipedia.org/wiki/Audi_TT_Mk3) (2014–2023)
* [Audi R8](https://en.wikipedia.org/wiki/Audi_R8#Second_generation_(2015%E2%80%93present:_Type_4S)) (2015–2023)
* [Bentley Flying Spur](https://en.wikipedia.org/wiki/Bentley_Flying_Spur_(2005)#Second_generation_(2013%E2%80%932019)) (2013-2019)
* [Bentley Mulsanne](https://en.wikipedia.org/wiki/Bentley_Mulsanne_(2010)) (2010–2020)[[5]](https://en.wikipedia.org/wiki/FlexRay#cite_note-:0-5)
* [BMW X5 (E70)](https://en.wikipedia.org/wiki/BMW_X5) (2006–2013)[[5]](https://en.wikipedia.org/wiki/FlexRay#cite_note-:0-5)
* [BMW X6 (E71)](https://en.wikipedia.org/wiki/BMW_X6) (2008–2014)[[10]](https://en.wikipedia.org/wiki/FlexRay#cite_note-10)
* [BMW 1 Series](https://en.wikipedia.org/wiki/BMW_F20)
* [BMW 3 Series](https://en.wikipedia.org/wiki/BMW_F30)
* [BMW 5 Series](https://en.wikipedia.org/wiki/BMW_5_Series_(F10)) (2009–2017)[[5]](https://en.wikipedia.org/wiki/FlexRay#cite_note-:0-5)
* [BMW 6 Series](https://en.wikipedia.org/wiki/BMW_6_Series) (2011–2018)[[11]](https://en.wikipedia.org/wiki/FlexRay#cite_note-11)
* [BMW 7 Series](https://en.wikipedia.org/wiki/BMW_F01) (2008–2015)[[5]](https://en.wikipedia.org/wiki/FlexRay#cite_note-:0-5)
* [Lamborghini Huracán](https://en.wikipedia.org/wiki/Lamborghini_Hurac%C3%A1n)
* [Mercedes-Benz S-Class (W222)](https://en.wikipedia.org/wiki/Mercedes-Benz_S-Class_(W222)) (2013–2020)[[12]](https://en.wikipedia.org/wiki/FlexRay#cite_note-12)
* [Mercedes-Benz S-Class (C217)](https://en.wikipedia.org/wiki/Mercedes-Benz_S-Class_(C217)) (2014–2020)[[13]](https://en.wikipedia.org/wiki/FlexRay#cite_note-13)
* [Mercedes-Benz E-Class (W213)](https://en.wikipedia.org/wiki/Mercedes-Benz_E-Class_(W213)) (2016–2023)
* [Mercedes-Benz C-Class (W205)](https://en.wikipedia.org/wiki/Mercedes-Benz_C-Class_(W205)) (2015-2023)
* [Mercedes-Benz C-Class (W206)](https://en.wikipedia.org/wiki/Mercedes-Benz_C-Class_(W206)) (2021–)
* [Mercedes-Benz S-Class (W223)](https://en.wikipedia.org/wiki/Mercedes-Benz_S-Class_(W223)) (2020–)
* [Rolls-Royce Ghost](https://en.wikipedia.org/wiki/Rolls-Royce_Ghost) (2009–)[[5]](https://en.wikipedia.org/wiki/FlexRay#cite_note-:0-5)
* [Land Rover](https://en.wikipedia.org/wiki/Land_Rover)
* [Volvo XC90](https://en.wikipedia.org/wiki/Volvo_XC90) (2015–)[[14]](https://en.wikipedia.org/wiki/FlexRay#cite_note-14)

**Details**

[[edit](https://en.wikipedia.org/w/index.php?title=FlexRay&action=edit&section=5)]

**Clock**

[[edit](https://en.wikipedia.org/w/index.php?title=FlexRay&action=edit&section=6)]

The FlexRay system consists of a bus and [electronic control units](https://en.wikipedia.org/wiki/Electronic_control_unit) (ECUs). Each ECU has an independent clock. The [clock drift](https://en.wikipedia.org/wiki/Clock_drift) must be not more than 0.15% from the reference clock, so the difference between the slowest and the fastest clock in the system is no greater than 0.3%.

This means that, if ECU-s is a sender and ECU-r is a receiver, then for every 300 cycles of the sender there will be between 299 and 301 cycles of the receiver. The clocks are resynchronized frequently enough to assure that this causes no problems. The clock is sent in the static segment.[[15]](https://en.wikipedia.org/wiki/FlexRay#cite_note-15)

**Bits on the bus**

[[edit](https://en.wikipedia.org/w/index.php?title=FlexRay&action=edit&section=7)]

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |   Correct averaging in case of no errors. The signal is merely delayed by 2 cycles. |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |   Errors near the middle of 8-cycle region are canceled. |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |   Errors near the boundary of 8-cycle region may affect the boundary bit. |

At each time, only one ECU writes to the [bus](https://en.wikipedia.org/wiki/Bus_(computing)). Each bit to be sent is held on the bus for 8 sample clock cycles. The receiver keeps a buffer of the last 5 samples, and uses the majority of the last 5 samples as the input signal.

Single-cycle transmission errors may affect results near the boundary of the bits, but will not affect cycles in the middle of the 8-cycle region.

**Sampled bits**

[[edit](https://en.wikipedia.org/w/index.php?title=FlexRay&action=edit&section=8)]

The value of the bit is sampled in the middle of the 8-bit region. The errors are moved to the extreme cycles, and the clock is synchronized frequently enough for the drift to be small. (Drift is smaller than 1 cycle per 300 cycles, and during transmission the clock is synchronized more than once every 300 cycles.)

**Frame**

[[edit](https://en.wikipedia.org/w/index.php?title=FlexRay&action=edit&section=9)]

All the communication is sent in the form of frames. The message consists of bytes {x0,x1,…,xm−1}, packed in the following way:

* Transmission Start Signal (TSS) – bit 0
* Frame Start Signal (FSS) – bit 1
* *m* times:
  + Byte Start Signal 0 (BSS0) – bit 1
  + Byte Start Signal 1 (BSS1) – bit 0
  + 0th bit of *i*-th byte
  + 1st bit of *i*-th byte
  + 2nd bit of *i*-th byte
  + ...
  + 7th bit of *i*-th byte
* Frame End Signal (FES) – bit 0
* Transmission End Signal (TES) – bit 1

If nothing is being communicated, the bus is held in state 1 (high voltage), so every receiver knows that the communication started when the voltage drops to 0.

The receiver knows when the message is complete by checking whether BSS0 (1) or FES (0) was received.

Note that 8-cycle per bit has nothing to do with bytes. Each byte takes 80 cycles to transfer. 16 for BSS0 and BSS1 and 64 for its bits. Also note that BSS0 has value 1, and BSS1 has value 0.

**Clock synchronization**

[[edit](https://en.wikipedia.org/w/index.php?title=FlexRay&action=edit&section=10)]

Clocks are resynchronized when the voted signal changes from 1 to 0, if the receiver was in either idle state or expecting BSS1.

As synchronization is done on the voted signal, small transmission errors during synchronization that affect the boundary bits may skew the synchronization no more than 1 cycle. As there are at most 88 cycles between synchronization (BSS1, 8 bits of the last byte, FES and TES - 11 bits of 8 cycles each), and the clock drift is no larger than 1 per 300 cycles, the drift may skew the clock no more than 1 cycle. Small transmission errors during the receiving may affect only the boundary bits. So in the worst case the two middle bits are correct, and thus the sampled value is correct.

Here's an example of a particularly bad case - error during synchronization, a lost cycle due to clock drift and error in transmission.

Errors that happened in the example:

* Because of a single-bit error during synchronization, the synchronization was delayed by 1 cycle
* Receiver clock was slower than sender clock, so receiver missed one cycle (marked X). This will not happen again before the next synchronization due to limits on maximum allowable clock drift.
* Because of a single-bit error during transmission, a bit was voted wrongly near the result.

Despite so many errors, the communication was received correctly.

The green cells are sampling points. All except the first are synchronized by the 1->0 edge in the transmission fragment shown.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Signal to be sent** | 1 | | | | | | | | 0 | | | | | | | | 1 | | | | | | | | 0 | | | | | | | | 1 | |
| **Signal sent** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| **On the bus** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| **Received** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | X | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| **5-maj voted** |  | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | X | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

**Development tools**

[[edit](https://en.wikipedia.org/w/index.php?title=FlexRay&action=edit&section=11)]

When developing and/or troubleshooting the FlexRay bus, examination of hardware signals can be very important. [Logic analyzers](https://en.wikipedia.org/wiki/Logic_analyzers) and [bus analyzers](https://en.wikipedia.org/wiki/Bus_analyzer) are tools which collect, analyze, decode, store signals so people can view the high-speed waveforms at their leisure.

**The future of FlexRay**

[[edit](https://en.wikipedia.org/w/index.php?title=FlexRay&action=edit&section=12)]

Ethernet may replace FlexRay for bandwidth intensive, non-safety critical applications.[[16]](https://en.wikipedia.org/wiki/FlexRay#cite_note-16)

---------------------------------------------------------------------------------------------------------------------

Когда говорят, что современные автомобили это компьютеры на колесах, вас не обманывают.

В какой-то мере это действительно так. Но под абстрактным словом компьютер скрываются вполне конкретные устройства, способные передать водителю информацию о состоянии отдельных узлов двигателя.

Конечно, здесь речь идет именно о датчиках, без которых сегодня, пожалуй, не выпускается ни одна машина.

Другое дело, что набор датчиков может отличаться не только от марки к марке авто, но и от их комплектации.

Поэтому мы расскажем о наиболее популярных и распространенных видах.

## Датчик массового расхода воздуха (ДМРВ)

Его функция учитывать количество поступающего воздуха в камеры сгорания смеси топлива.

Устанавливается, как правило, перед дроссельной заслонкой после воздушного фильтра.

В разных автомобилях этот тип датчиков может быть представлен в проволочном, мембранном или объемном исполнении, при этом его функции отличаться не будут.

Проблемы с ним выражаются в повышенном расходе топлива, затрудненный запуск двигателя, низкие обороты на холостом ходу,  
падение мощности двигателя.

Кстати, это самый дорогой датчик из нашего списка. Цена даже для отечественных автомобилей в районе 5 тысяч рублей.



Датчик положения коленчатого вала (ДПКВ)

Если он выйдет из строя, то автомобиль не получится эксплуатировать до устранения неполадки. Пожалуй, в этом и состоит его существенное отличие от остальных видов.

Он выполняет одну архи важную функцию фиксирует положение коленчатого вала и передает информацию на ЭБУ (тот самый абстрактный компьютер). В свою очередь ЭБУ корректирует объем смеси топлива, поступающего в двигатель, время подачи этого топлива, угол опережения зажигания и угол поворота распредвала.

Как вы понимаете, если датчик, то сегодня вы едете на такси

Определить неисправность легко: двигатель или не запускается совсем, или сразу глохнет в первые секунды. На ранней стадии – отсутствие холостого хода.



## Датчик температуры охлаждающей жидкости (ДТОЖ)

Его можно сравнить с природным газом. Странно, не так ли? Возможно, но газ в необработанном виде не имеет цвета и запаха, а значит, вдыхая его, человек даже не заметит, как постепенно будет умирать. Неисправный ДТОЖ не передаст информацию в ЭБУ, а он не запустит вентилятор, когда двигатель нужно будет остужать. Итог печален и урон несоразмерен со стоимостью нового датчика.

Как определить? Только по перегретому двигателю: рывки при ускорении, затрудненный запуск, завышенные обороты на холостом ходу, снижение мощности.



## Датчик положения дроссельной заслонки (ДПДЗ)

Это небольшое устройство контролирует объем подачи топлива, фиксируя положение дроссельной заслонки. В зависимости от угла поворота заслонки меняется напряжение и передается на ЭБУ. Находится на корпусе дроссельной заслонки.

С высокой долей вероятности, определить неисправность не составит труда: неровный холостой ход, затрудненный запуск ДВС, заметное падение оборотов после перегазовки, обороты зависают после нажатия на педаль газа.

Понаблюдайте за своим автомобилем. Возможно, ДПДЗ скоро окончательно выйдет из строя.



## Регулятор холостого хода (РХХ)

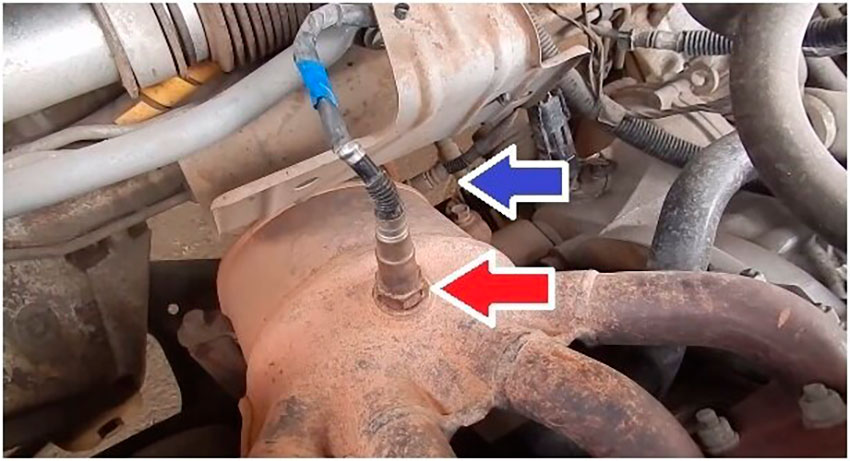
Как следует из названия, этот датчик обеспечивает работу двигателя в тот момент, когда дроссельная заслонка закрыта. Он же отвечает за работу двигателя в прогретом состоянии и позволяет начать движение, не дожидаясь доведения ДВС до рабочих температур.

Если с ним что-то не так, то автомобиль может запускаться только при нажатии педали газа.



## Датчик кислорода (лямбда зонд)

Расположен на приемной трубе глушителя. Его задача анализировать состав отработанных газов и определять уровень несгоревшего кислорода (если таковой вообще есть). При наличии избыточного кислорода бедная топливная смесь, при отсутствии богатая. Показания датчика кислорода используются для корректировки подачи топлива.



Признаки неисправности: запоздалая реакция двигателя на педаль газа, снижение мощности, повышенный расход топлива.

## Датчик детонации

Нужен для того, чтобы ЭБУ выставлял корректный уровень опережения зажигания. Установленный в блоке цилиндров, он получает данные из камер сгорания.

Признаки неисправности классические – потеря мощности, увеличенный расход топлива и повышенная температура двигателя.



## Выводы

Как вы обратили внимание, вне зависимости от типа датчика, его поломка или постепенный выход из строя сопровождается схожими последствиями потерей мощности, повышенным расходом топлива, проблемами с запуском.

Если вы сталкиваетесь со схожими трудностями, самое время посетить сервис и провести компьютерную диагностику на ней проблемы с датчиками идентифицируются вполне отчетливо.

----------------------------------------------------------------------------------------------------------------

**On-board diagnostics** (**OBD**) is a term referring to a vehicle's self-diagnostic and reporting capability. In the United States, this capability is a requirement to comply with [federal emissions standards](https://en.wikipedia.org/wiki/United_States_vehicle_emission_standards) to detect failures that may increase the vehicle tailpipe [emissions](https://en.wikipedia.org/wiki/Air_pollution) to more than 150% of the standard to which it was originally certified.[[1]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-1)[[2]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-2)

OBD systems give the vehicle owner or repair technician access to the status of the various vehicle sub-systems. The amount of diagnostic information available via OBD has varied widely since its introduction in the early 1980s versions of onboard vehicle computers. Early versions of OBD would simply illuminate a [tell-tale](https://en.wikipedia.org/wiki/Tell-tale_(automotive)) light if a problem was detected, but would not provide any information as to the nature of the problem. Modern OBD implementations use a standardized digital communications port to provide [real-time data](https://en.wikipedia.org/wiki/Real-time_data) and [diagnostic trouble codes](https://en.wikipedia.org/wiki/Table_of_OBD-II_Codes) which allow malfunctions within the vehicle to be rapidly identified.

**History**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=1)]

|  |  |
| --- | --- |
| https://upload.wikimedia.org/wikipedia/en/thumb/f/f2/Edit-clear.svg/40px-Edit-clear.svg.png | This section **is in**[**list**](https://en.wikipedia.org/wiki/MOS:LIST)**format but may read better as**[**prose**](https://en.wikipedia.org/wiki/MOS:PROSE). You can help by [converting this section](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit), if appropriate. [Editing help](https://en.wikipedia.org/wiki/Help:Editing) is available. *(September 2021)* |

* 1968: [Volkswagen](https://en.wikipedia.org/wiki/Volkswagen) introduces the first on-board computer system, in their [fuel-injected](https://en.wikipedia.org/wiki/Fuel-injected) [Type 3](https://en.wikipedia.org/wiki/Volkswagen_Type_3) models. This system is entirely analog with no diagnostic capabilities.
* 1975: Bosch and Bendix EFI systems are adopted by major automotive manufacturers to improve tailpipe (exhaust) emissions. These systems are also analog, though some provide rudimentary diagnostic capability through factory tools, such as the Kent Moore J-25400, compatible with the [Datsun 280Z](https://en.wikipedia.org/wiki/Datsun_280Z), and the [Cadillac Seville.](https://en.wikipedia.org/wiki/Cadillac_Seville#First_generation_(1976%E2%80%931979))
* 1980: [General Motors](https://en.wikipedia.org/wiki/General_Motors) introduces the first data link on their 1980 [Cadillac Eldorado](https://en.wikipedia.org/wiki/Cadillac_Eldorado) and Seville models. Diagnostic Trouble Codes (DTCs) are displayed through the electronic climate control system's digital readout when in diagnostic mode.[[3]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-3)
* 1981: [General Motors](https://en.wikipedia.org/wiki/General_Motors) introduced its "Computer Command Control" system on all US passenger vehicles for model year 1981. Included in this system is a proprietary 5-pin [ALDL](https://en.wikipedia.org/wiki/ALDL) that interfaces with the [Engine Control Module](https://en.wikipedia.org/wiki/Engine_Control_Module) (ECM) to initiate a diagnostic request and provide a serial data stream. The protocol communicates at 160 [baud](https://en.wikipedia.org/wiki/Baud) with [Pulse-width modulation](https://en.wikipedia.org/wiki/Pulse-width_modulation) (PWM) signaling and monitors all engine management functions. It reports real-time sensor data, component overrides, and Diagnostic Trouble Codes. The specification for this link is as defined by GM's Emissions Control System Project Center document XDE-5024B.[[4]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-4)[[5]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-5)
* 1982: [RCA](https://en.wikipedia.org/wiki/RCA) defines an analog STE/ICE (simplified test equipment for internal combustion engines) vehicle diagnostic standard used in the [CUCV](https://en.wikipedia.org/wiki/Commercial_Utility_Cargo_Vehicle), [M60 tank](https://en.wikipedia.org/wiki/M60_tank) and other military vehicles of the era for the US Army.[[6]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-6)
* 1986: [General Motors](https://en.wikipedia.org/wiki/General_Motors) introduces an upgraded version of the ALDL protocol, which communicates at 8192 baud with half-duplex [UART](https://en.wikipedia.org/wiki/UART) signaling on some models.
* 1988: The [California Air Resources Board](https://en.wikipedia.org/wiki/California_Air_Resources_Board) (CARB) requires that all new vehicles sold in [California](https://en.wikipedia.org/wiki/California) from 1988 onward have some basic OBD capability (such as detecting problems with fuel metering and [Exhaust gas recirculation](https://en.wikipedia.org/wiki/Exhaust_gas_recirculation).)[[7]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-7)[[8]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-Ford_manual-8) These requirements are generally referred to as "OBD-I", though this name is a [retronym](https://en.wikipedia.org/wiki/Retronym) applied after the introduction of OBD-II. The data link connector and its position are not standardized, nor is the data protocol. The [Society of Automotive Engineers (SAE)](https://en.wikipedia.org/wiki/SAE_International) recommends a standardized diagnostic connector and set of diagnostic test signals.
* ~1994: Motivated by a desire for a state-wide [emissions testing](https://en.wikipedia.org/wiki/Automobile_emissions_control) program, the CARB issues the OBD-II specification and mandates that it be adopted for all cars sold in California starting in model year 1996 (see CCR Title 13 Section 1968.1 and 40 CFR Part 86 Section 86.094). The DTCs and connectors suggested by the SAE are incorporated into this specification.
* 1996: The OBD-II specification is made mandatory for all passenger cars and petrol-powered light trucks with a [gross vehicle weight rating](https://en.wikipedia.org/wiki/Vehicle_weight#Gross_vehicle_weight_rating) less than 8,500 lb (3,900 kg) in the United States. The OBD-II specification is also made mandatory for all petrol-powered vehicles with [California emissions](https://en.wikipedia.org/wiki/California_emission_standards) with a gross vehicle weight rating up to 14,000 lb (6,400 kg).[[8]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-Ford_manual-8)
* 1997: The OBD-II specification is made mandatory for California emissions diesel-engined vehicles with a gross vehicle weight rating up to 14,000 lb (6,400 kg).[[8]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-Ford_manual-8)
* 2001: The [European Union](https://en.wikipedia.org/wiki/European_Union) makes [EOBD](https://en.wikipedia.org/wiki/On-board_diagnostics#EOBD) mandatory for all petrol vehicles sold in the European Union, starting in MY2001 (see [European emission standards](https://en.wikipedia.org/wiki/European_emission_standards) Directive 98/69/EC[[9]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-9)).
* 2004: The [European Union](https://en.wikipedia.org/wiki/European_Union) makes [EOBD](https://en.wikipedia.org/wiki/On-board_diagnostics#EOBD) mandatory for all diesel vehicles sold in the European Union. All petrol-powered vehicles in the United States with a gross vehicle weight rating of up to 14,000 lb (6,400 kg) are required to have OBD-II.[[8]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-Ford_manual-8)
* 2006: All vehicles manufactured in [Australia](https://en.wikipedia.org/wiki/Australia) and [New Zealand](https://en.wikipedia.org/wiki/New_Zealand) are required to be OBD-II compliant after January 1, 2006.[[10]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-10) All vehicles in the United States of 14,000 lb (6,400 kg) gross vehicle weight rating and under are required to have OBD-II.[[8]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-Ford_manual-8)
* 2007: All California emissions vehicles over 14,000 lb (6,400 kg) gross vehicle weight rating are required to support EMD/EMD+ or OBD-II.
* 2008: All cars sold in the United States are required to use the [ISO 15765-4](https://en.wikipedia.org/w/index.php?title=ISO_15765-4&action=edit&redlink=1)[[11]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-11) signaling standard (a variant of the [Controller Area Network](https://en.wikipedia.org/wiki/Controller_Area_Network) (CAN) [bus](https://en.wikipedia.org/wiki/Bus_(computing))).[[12]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-12)
* 2008: Certain light vehicles in China are required by the Environmental Protection Administration Office to implement OBD (standard GB18352[[13]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-13)) by July 1, 2008.[[14]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-14) Some regional exemptions may apply.
* 2010: Start of required phase-in of the OBD-II specification to all vehicles with a gross vehicle weight rating of 14,000 lb (6,400 kg) and above, this was completed by the 2013 model year. Vehicles that did not have OBD-II during this time period were required to have EMD/EMD+.[[8]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-Ford_manual-8)

**Standard interfaces**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=2)]

**ALDL**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=3)]

*Main article:*[*ALDL*](https://en.wikipedia.org/wiki/ALDL)

GM's [ALDL](https://en.wikipedia.org/wiki/ALDL) (Assembly Line Diagnostic Link) is sometimes referred to as a predecessor to, or a manufacturer's proprietary version of, an OBD-I diagnostic starting in 1981. This interface was made in different varieties and changed with power train control modules (aka PCM, ECM, ECU). Different versions had slight differences in pin-outs and baud rates. Earlier versions used a 160 baud rate, while later versions went up to 8192 baud and used bi-directional communications to the PCM.[[15]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-15)[[16]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-16)

**OBD-I**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=4)]

The regulatory intent of OBD-I was to encourage auto manufacturers to design reliable [emission control systems](https://en.wikipedia.org/wiki/Emission_control_system) that remain effective for the vehicle's "useful life".[[17]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-17) The hope was that by forcing annual emissions testing for [California](https://en.wikipedia.org/wiki/California) starting in 1988, [[18]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-18) and denying registration to vehicles that did not pass, drivers would tend to purchase vehicles that would more reliably pass the test. OBD-I was largely unsuccessful, as the means of reporting emissions-specific diagnostic information was not standardized. Technical difficulties with obtaining standardized and reliable emissions information from all vehicles led to an inability to implement the annual testing program effectively.[[19]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-19)

The Diagnostic Trouble Codes (DTC's) of OBD-I vehicles can usually be found without an expensive scan tool. Each manufacturer used their own Diagnostic Link Connector (DLC), DLC location, DTC definitions, and procedure to read the DTC's from the vehicle. DTC's from OBD-I cars are often read through the blinking patterns of the 'Check Engine Light' (CEL) or 'Service Engine Soon' (SES) light. By connecting certain pins of the diagnostic connector, the 'Check Engine' light will blink out a two-digit number that corresponds to a specific error condition. The DTC's of some OBD-I cars are interpreted in different ways, however. Cadillac fuel-injected vehicles are equipped with actual *onboard* diagnostics, providing trouble codes, actuator tests and sensor data through the new digital Electronic Climate Control display.

Holding down 'Off' and 'Warmer' for several seconds activates the diagnostic mode without the need for an external scan tool. Some Honda engine computers are equipped with [LEDs](https://en.wikipedia.org/wiki/Light-emitting_diode) that light up in a specific pattern to indicate the DTC. General Motors, some 1989–1995 Ford vehicles (DCL), and some 1989–1995 Toyota/Lexus vehicles have a live sensor data stream available; however, many other OBD-I equipped vehicles do not. OBD-I vehicles have fewer DTC's available than OBD-II equipped vehicles.

**OBD-1.5**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=5)]

OBD 1.5 refers to a partial implementation of OBD-II which [General Motors](https://en.wikipedia.org/wiki/General_Motors) used on some vehicles in 1994, 1995, & 1996. (GM did not use the term OBD 1.5 in the documentation for these vehicles — they simply have an OBD and an OBD-II section in the service manual.)

For example, the 1994–1995 model year Corvettes have one post-catalyst [oxygen sensor](https://en.wikipedia.org/wiki/Oxygen_sensor) (although they have two [catalytic converters](https://en.wikipedia.org/wiki/Catalytic_converter)), and have a subset of the OBD-II codes implemented.[[20]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-20)

This hybrid system was present on GM [B-body](https://en.wikipedia.org/wiki/General_Motors_B_platform) cars (the Chevrolet Caprice, Impala, and Buick Roadmaster) for 1994–1995model years, [H-body](https://en.wikipedia.org/wiki/General_Motors_H_platform_(1986)) cars for 1994–1995, [W-body](https://en.wikipedia.org/wiki/General_Motors_W_platform) cars (Buick Regal, Chevrolet Lumina (for 1995 only), Chevrolet Monte Carlo (1995 only), Pontiac Grand Prix, Oldsmobile Cutlass Supreme) for 1994–1995, [L-body](https://en.wikipedia.org/wiki/General_Motors_L_platform) (Chevrolet Beretta/Corsica) for 1994–1995, [Y-body](https://en.wikipedia.org/wiki/General_Motors_Y_platform) (Chevrolet Corvette) for 1994–1995, on the [F-body](https://en.wikipedia.org/wiki/General_Motors_F_platform) (Chevrolet Camaro and Pontiac Firebird) for 1995 and on the [J-Body](https://en.wikipedia.org/wiki/General_Motors_J_platform) (Chevrolet Cavalier and Pontiac Sunfire) and [N-Body](https://en.wikipedia.org/wiki/General_Motors_N_platform) (Buick Skylark, Oldsmobile Achieva, Pontiac Grand Am) for 1995 and 1996 and also for North American delivered 1994–1995 Saab vehicles with the naturally aspirated 2.3.

The pinout for the ALDL connection on these cars is as follows:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |

For ALDL connections, pin 9 is the data stream, pins 4 and 5 are ground, and pin 16 is the battery voltage.

An OBD 1.5 compatible scan tool is required to read codes generated by OBD 1.5.

Additional vehicle-specific diagnostic and control circuits are also available on this connector. For instance, on the Corvette there are interfaces for the Class 2 serial data stream from the PCM, the CCM diagnostic terminal, the radio data stream, the airbag system, the selective ride control system, the low tire pressure warning system, and the passive keyless entry system.[[21]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-21)

An OBD 1.5 has also been used in the Ford Scorpio since 95.[[22]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-22)

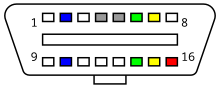
**OBD-II**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=6)]

OBD-II is an improvement over OBD-I in both capability and standardization. The OBD-II standard specifies the type of diagnostic connector and its pinout, the electrical signalling protocols available, and the messaging format. It also provides a candidate list of vehicle parameters to monitor along with how to encode the data for each. There is a pin in the connector that provides power for the scan tool from the vehicle battery, which eliminates the need to connect a scan tool to a power source separately. However, some technicians might still connect the scan tool to an auxiliary power source to protect data in the unusual event that a vehicle experiences a loss of electrical power due to a malfunction. Finally, the OBD-II standard provides an extensible list of DTCs. As a result of this standardization, a single device can query the on-board computer(s) in any vehicle. This OBD-II came in two models OBD-IIA and OBD-IIB. OBD-II standardization was prompted by emissions requirements, and though only emission-related codes and data are required to be transmitted through it, most manufacturers have made the OBD-II [Data Link Connector](https://en.wikipedia.org/wiki/Data_link_connector_(automotive)) the only one in the vehicle through which all systems are diagnosed and programmed. OBD-II Diagnostic Trouble Codes are 4-digit, preceded by a letter: P for powertrain (engine and transmission), B for body, C for chassis, and U for network.

**OBD-II diagnostic connector**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=7)]

[](https://en.wikipedia.org/wiki/File:OBD_002.jpg)Female OBD-II connector on a car[](https://en.wikipedia.org/wiki/File:OBD-II_type_A_female_connector_pinout.svg)Female OBD-II type A connector pinout – front view[](https://en.wikipedia.org/wiki/File:OBD-II_type_B_female_connector_pinout.svg)Female OBD-II type B connector pinout – front view. Wire placement is identical to type A, but the center groove is split in two.

The OBD-II specification provides for a standardized hardware interface — the female 16-pin (2x8) [J1962 connector](https://en.wikipedia.org/wiki/Data_link_connector_(automotive)#OBD-II_diagnostic_connector), where type A is used for 12-volt vehicles and type B for 24-volt vehicles. Unlike the OBD-I connector, which was sometimes found under the bonnet of the vehicle, the OBD-II connector is required to be within 2 feet (0.61 m) of the steering wheel (unless an exemption is applied for by the manufacturer, in which case it is still somewhere within reach of the driver).

SAE J1962 defines the pinout of the connector as:

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **Manufacturer discretion**  GM: J2411 GMLAN/SWC/Single-Wire CAN. Audi: Switched +12 to tell a scan tool whether the ignition is on. VW: Switched +12 to tell a scan tool whether the ignition is on. Mercedes[[23]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-23) (K-Line): Ignition control (EZS), air-conditioner (KLA), PTS, safety systems (Airbag, SRS, AB) and some other. | **9** | **Manufacturer discretion**  GM: 8192 baud ALDL where fitted. BMW: RPM signal. Toyota: RPM signal. Mercedes (K-Line): ABS, ASR, ESP, ETS, BAS diagnostic. |
| **2** | [**Bus**](https://en.wikipedia.org/wiki/Bus_(computing))**positive Line**  [SAE J1850](https://en.wikipedia.org/wiki/SAE_J1850) PWM and VPW | **10** | **Bus negative Line**  SAE J1850 PWM only (not SAE 1850 VPW) |
| **3** | **Manufacturer discretion**  Ethernet TX+ (Diagnostics over IP) Ford DCL(+) Argentina, Brazil (pre OBD-II) 1997–2000, USA, Europe, etc. Chrysler CCD Bus(+) Mercedes (TNA): TD engine rotation speed. | **11** | **Manufacturer discretion**  Ethernet TX- (Diagnostics over IP) Ford DCL(-) Argentina, Brazil (pre OBD-II) 1997–2000, USA, Europe, etc. Chrysler CCD Bus(-) Mercedes (K-Line): Gearbox and other transmission components (EGS, ETC, FTC). |
| **4** | **Chassis ground** | **12** | **Manufacturer discretion**  Ethernet RX+ (Diagnostics over IP) Mercedes (K-Line): All activity module (AAM), Radio (RD), ICS (and more) |
| **5** | **Signal ground** | **13** | **Manufacturer discretion**  Ethernet RX- (Diagnostics over IP) Ford: FEPS – Programming PCM voltage Mercedes (K-Line): AB diagnostic – safety systems. |
| **6** | **CAN high**  (ISO 15765-4 and SAE J2284) | **14** | **CAN low**  (ISO 15765-4 and SAE J2284) |
| **7** | **K-line**  (ISO 9141-2 and ISO 14230-4) | **15** | **L-line**  (ISO 9141-2 and ISO 14230-4) |
| **8** | **Manufacturer discretion**  Activate Ethernet (Diagnostics over IP) Many BMWs: A second K-line for non OBD-II (Body/Chassis/Infotainment) systems. Mercedes: Ignition | **16** | **Battery voltage**  (+12 Volt for type A connector) (+24 Volt for type B connector) |

The assignment of unspecified pins is left to the vehicle manufacturer's discretion.[[24]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-24)

**EOBD**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=8)]

The European on-board diagnostics (EOBD) regulations are the European equivalent of OBD-II, and apply to all passenger cars of category M1 (with no more than 8 passenger seats and a Gross Vehicle Weight rating of 2,500 kg, 5,500 lb or less) first registered within EU member states since January 1, 2001 for [petrol](https://en.wikipedia.org/wiki/Petrol)-engined cars and since January 1, 2004 for [diesel](https://en.wikipedia.org/wiki/Diesel_fuel) engined cars.[[25]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-25)

For newly introduced models, the regulation dates applied a year earlier – January 1, 2000 for petrol and January 1, 2003, for diesel.  
For passenger cars with a Gross Vehicle Weight rating of greater than 2500 kg and for light commercial vehicles, the regulation dates applied from January 1, 2002, for petrol models, and January 1, 2007, for diesel models.

The technical implementation of EOBD is essentially the same as OBD-II, with the same SAE J1962 diagnostic link connector and signal protocols being used.

With Euro V and Euro VI emission standards, EOBD emission thresholds are lower than previous Euro III and IV.

**EOBD fault codes**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=9)]

Each of the EOBD fault codes consists of five characters: a letter, followed by four numbers.[[26]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-26) The letter refers to the system being interrogated e.g. Pxxxx would refer to the powertrain system. The next character would be a 0 if complies to the EOBD standard. So it should look like P0xxx.

The next character would refer to the sub system.

* P00xx – Fuel and Air Metering and Auxiliary Emission Controls.
* P01xx – Fuel and Air Metering.
* P02xx – Fuel and Air Metering (Injector Circuit).
* P03xx – Ignition System or Misfire.
* P04xx – Auxiliary Emissions Controls.
* P05xx – Vehicle Speed Controls and Idle Control System.
* P06xx – Computer Output Circuit.
* P07xx – Transmission.
* P08xx – Transmission.

The following two characters would refer to the individual fault within each subsystem.[[27]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-27)

**EOBD2**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=10)]

The term "EOBD2" is [marketing speak](https://en.wikipedia.org/wiki/Marketing_speak) used by some vehicle manufacturers to refer to manufacturer-specific features that are not actually part of the OBD or EOBD standard. In this case "E" stands for Enhanced.

**JOBD**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=11)]

JOBD is a version of OBD-II for vehicles sold in Japan.

**ADR 79/01 & 79/02 (Australian OBD standard)**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=12)]

The ADR 79/01 (Vehicle Standard (**A**ustralian **D**esign **R**ule **79/01** – Emission Control for Light Vehicles) 2005) standard is the Australian equivalent of OBD-II. It applies to all vehicles of category M1 and N1 with a Gross Vehicle Weight rating of 3,500 kg (7,700 lb) or less, registered from new within Australia and produced since January 1, 2006 for [petrol](https://en.wikipedia.org/wiki/Petrol)-engined cars and since January 1, 2007 for [diesel](https://en.wikipedia.org/wiki/Diesel_fuel)-engined cars.[[28]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-28)

For newly introduced models, the regulation dates applied a year earlier – January 1, 2005 for petrol and January 1, 2006, for diesel. The ADR 79/01 standard was supplemented by the ADR 79/02 standard which imposed tighter emissions restrictions, applicable to all vehicles of class M1 and N1 with a Gross Vehicle Weight rating of 3500 kg or less, from July 1, 2008, for new models, July 1, 2010, for all models.[[29]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-29)

The technical implementation of this standard is essentially the same as OBD-II, with the same SAE J1962 diagnostic link connector and signal protocols being used.

**EMD/EMD+**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=13)]

In North America, EMD and EMD+ are on-board diagnostic systems that were used on vehicles with a gross vehicle weight rating of 14,000 lb (6,400 kg) or more between the 2007 and 2012 model years if those vehicles did not already implement OBD-II. EMD was used on California emissions vehicles between model years 2007 and 2009 that did not already have OBD-II. EMD was required to monitor fuel delivery, exhaust gas recirculation, the [diesel particulate filter](https://en.wikipedia.org/wiki/Diesel_particulate_filter) (on diesel engines), and emissions-related powertrain control module inputs and outputs for circuit continuity, data rationality, and output functionality. EMD+ was used on model year 2010-2012 California and Federal petrol-engined vehicles with a gross vehicle weight rating of over 14,000 lb (6,400 kg), it added the ability to monitor nitrogen oxide catalyst performance. EMD and EMD+ are similar to OBD-I in logic but use the same SAE J1962 data connector and CAN bus as OBD-II systems.[[8]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-Ford_manual-8)

**OBD-II signal protocols**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=14)]

Five signaling protocols are permitted with the OBD-II interface. Most vehicles implement only one of the protocols. It is often possible to deduce the protocol used based on which pins are present on the J1962 connector:[[30]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-30)

* SAE J1850 PWM ([pulse-width modulation](https://en.wikipedia.org/wiki/Pulse-width_modulation) — 41.6 kB/sec, standard of the [Ford Motor Company](https://en.wikipedia.org/wiki/Ford_Motor_Company))
  + pin 2: Bus+
  + pin 10: Bus–
  + High voltage is +5 V
  + Message length is restricted to 12 bytes, including [CRC](https://en.wikipedia.org/wiki/Cyclic_redundancy_check)
  + Employs a multi-master arbitration scheme called '[Carrier Sense Multiple Access](https://en.wikipedia.org/wiki/Carrier_Sense_Multiple_Access) with Non-Destructive Arbitration' (CSMA/NDA)
* SAE J1850 VPW ([variable pulse width](https://en.wikipedia.org/w/index.php?title=Variable_pulse_width&action=edit&redlink=1) — 10.4/41.6 kB/sec, standard of [General Motors](https://en.wikipedia.org/wiki/General_Motors_Corporation))
  + pin 2: Bus+
  + Bus idles low
  + High voltage is +7 V
  + Decision point is +3.5 V
  + Message length is restricted to 12 bytes, including CRC
  + Employs [CSMA](https://en.wikipedia.org/wiki/Carrier_Sense_Multiple_Access)/NDA
* [ISO 9141-2](https://en.wikipedia.org/w/index.php?title=ISO_9141-2&action=edit&redlink=1).[[31]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-31) This protocol has an asynchronous serial data rate of 10.4 kbit/s.[[32]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-32) It is somewhat similar to [RS-232](https://en.wikipedia.org/wiki/RS-232); however, the signal levels are different, and communications happen on a single, bidirectional line without additional handshake signals. ISO 9141-2 is primarily used in Chrysler, European, and Asian vehicles.
  + pin 7: K-line
  + pin 15: L-line (optional)
  + UART signaling
  + K-line idles high, with a 510 ohm resistor to Vbatt
  + The active/dominant state is driven low with an open-collector driver.
  + Message length is Max 260Bytes. Data field MAX 255.
* [ISO 14230](https://en.wikipedia.org/wiki/ISO_14230) KWP2000 ([Keyword Protocol 2000](https://en.wikipedia.org/wiki/Keyword_Protocol_2000))
  + pin 7: K-line
  + pin 15: L-line (optional)
  + Physical layer identical to ISO 9141-2
  + Data rate 1.2 to 10.4 kBaud
  + Message may contain up to 255 bytes in the data field
* ISO 15765 [CAN](https://en.wikipedia.org/wiki/Controller_Area_Network) (250 kbit/s or 500 kbit/s). The CAN protocol was developed by Bosch for automotive and industrial control. Unlike other OBD protocols, variants are widely used outside of the automotive industry. While it did not meet the OBD-II requirements for U.S. vehicles prior to 2003, as of 2008 all vehicles sold in the US are required to implement CAN as one of their signaling protocols.
  + pin 6: CAN High
  + pin 14: CAN Low

All OBD-II pinouts use the same connector, but different pins are used with the exception of pin 4 (battery ground) and pin 16 (battery positive).

**OBD-II diagnostic data available**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=15)]

OBD-II provides access to data from the [engine control unit](https://en.wikipedia.org/wiki/Engine_control_unit) (ECU) and offers a valuable source of information when troubleshooting problems inside a vehicle. The SAE J1979 standard defines a method for requesting various diagnostic data and a list of standard parameters that might be available from the ECU. The various available parameters are addressed by "parameter identification numbers" or **PID**s which are defined in J1979. For a list of basic PIDs, their definitions, and the formula to convert raw OBD-II output to meaningful diagnostic units, see [OBD-II PIDs](https://en.wikipedia.org/wiki/OBD-II_PIDs). Manufacturers are not required to implement all PIDs listed in J1979 and they are allowed to include proprietary PIDs that are not listed. The PID request and data retrieval system gives access to real time performance data as well as flagged DTCs. For a list of generic OBD-II DTCs suggested by the SAE, see [Table of OBD-II Codes](https://en.wikipedia.org/wiki/Table_of_OBD-II_Codes). Individual manufacturers often enhance the OBD-II code set with additional proprietary DTCs.

**Mode of operation/OBD services**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=16)]

Here is a basic introduction to the OBD [communication protocol](https://en.wikipedia.org/wiki/Communication_protocol) according to ISO 15031. In SAE J1979 these "modes" were renamed to "services", starting in 2003.

* **Service / Mode $01** shows current sensor live data from PIDs ("Parameter IDs"). See [OBD-II PIDs#Service\_01](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_01) for an extensive list.
* **Service / Mode $02** makes Freeze Frame data accessible via the same PIDs.[[33]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-33) See [OBD-II PIDs#Service\_02](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_02) for a list.
* **Service / Mode $03** lists the emission-related "confirmed" diagnostic trouble codes stored. It either displays numeric, 4 digit codes identifying the faults or maps them to a letter (P, B, U, C) plus 4 digits. See [#OBD-II\_diagnostic\_trouble\_codes](https://en.wikipedia.org/wiki/On-board_diagnostics#OBD-II_diagnostic_trouble_codes).
* **Service / Mode $04** is used to clear emission-related diagnostic information. This includes clearing the stored pending/confirmed DTCs and Freeze Frame data.[[34]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-34)
* **Service / Mode $05** displays the oxygen sensor monitor screen and the test results gathered about the oxygen sensor. There are ten numbers available for diagnostics:
  + $01 Rich-to-Lean O2 sensor threshold voltage
  + $02 Lean-to-Rich O2 sensor threshold voltage
  + $03 Low sensor voltage threshold for switch time measurement
  + $04 High sensor voltage threshold for switch time measurement
  + $05 Rich-to-Lean switch time in ms
  + $06 Lean-to Rich switch time in ms
  + $07 Minimum voltage for test
  + $08 Maximum voltage for test
  + $09 Time between voltage transitions in ms
  + See [OBD-II PIDs#Service\_05](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_05) for a list.
* **Service / Mode $06** is a Request for On-Board Monitoring Test Results for Continuously and Non-Continuously Monitored System. There are typically a minimum value, a maximum value, and a current value for each non-continuous monitor.
* **Service / Mode $07** is a Request for emission-related diagnostic trouble codes detected during current or last completed driving cycle. It enables the external test equipment to obtain "pending" diagnostic trouble codes detected during current or last completed driving cycle for emission-related components/systems. This is used by service technicians after a vehicle repair, and after clearing diagnostic information to see test results after a single driving cycle to determine if the repair has fixed the problem. See [#OBD-II\_diagnostic\_trouble\_codes](https://en.wikipedia.org/wiki/On-board_diagnostics#OBD-II_diagnostic_trouble_codes).
* **Service / Mode $08** could enable the off-board test device to control the operation of an on-board system, test, or component.
* **Service / Mode $09** is used to retrieve vehicle information. Among others, the following information is available:
  + VIN ([Vehicle Identification Number](https://en.wikipedia.org/wiki/Vehicle_identification_number)): Vehicle ID
  + CALID (Calibration Identification): ID for the software installed on the ECU
  + CVN (Calibration Verification Number): Number used to verify the integrity of the vehicle software. The manufacturer is responsible for determining the method of calculating CVN(s), e.g. using checksum.
  + In-use performance counters
    - Petrol engine : Catalyst, Primary oxygen sensor, Evaporating system, EGR system, VVT system, Secondary air system, and Secondary oxygen sensor
    - Diesel engine : NMHC catalyst, NOx reduction catalyst, NOx absorber Particulate matter filter, Exhaust gas sensor, EGR system, VVT system, Boost pressure control, Fuel system.
  + See [OBD-II PIDs#Service\_09](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_09) for an extensive list.
* **Service / Mode $0A** lists emission-related "permanent" diagnostic trouble codes stored. As per CARB, any diagnostic trouble codes that is commanding MIL on and stored into non-volatile memory shall be logged as a permanent fault code. See [#OBD-II\_diagnostic\_trouble\_codes](https://en.wikipedia.org/wiki/On-board_diagnostics#OBD-II_diagnostic_trouble_codes).

**Applications**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=17)]

Various tools are available that plug into the OBD connector to access OBD functions. These range from simple generic consumer level tools to highly sophisticated [OEM](https://en.wikipedia.org/wiki/Original_equipment_manufacturer) dealership tools to vehicle telematic devices.

**Hand-held scan tools**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=18)]

[](https://en.wikipedia.org/wiki/File:Autoboss_v-30.png)Multi-brand vehicle diagnostics system handheld Autoboss V-30 with adapters for connectors of several vehicle manufacturers.[[35]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-35)

A range of rugged hand-held scan tools is available.

* Simple fault code readers/reset tools are mostly aimed at the consumer level.
* Professional hand-held scan tools may possess more advanced functions
  + Access more advanced diagnostics
  + Set manufacturer- or vehicle-specific ECU parameters
  + Access and control other control units, such as air bag or ABS
  + Real-time monitoring or graphing of engine parameters to facilitate diagnosis or tuning

**Mobile device-based tools and analysis**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=19)]

Mobile device applications allow mobile devices such as cell phones and tablets to display and manipulate the OBD-II data accessed via [USB](https://en.wikipedia.org/wiki/USB) adaptor cables or [Bluetooth](https://en.wikipedia.org/wiki/Bluetooth) adapters plugged into the car's OBD II connector. Newer devices on the market are equipped with GPS sensors and the ability to transmit vehicle location and diagnostics data over a cellular network. Modern OBD-II devices can therefore nowadays be used to for example locate vehicles, monitor driving behavior in addition to reading Diagnostics Trouble Codes (DTC). Even more advanced devices allow users to reset engine DTC codes, effectively turning off engine lights in the dashboard; however, resetting the codes does not address the underlying issues and can in worst-case scenarios even lead to engine breakage where the source issue is serious and left unattended for long periods.[[36]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-36)[[37]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-37)

**OBD-II Software**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=20)]

An OBD-II software package when installed in a computer ([Windows](https://en.wikipedia.org/wiki/Microsoft_Windows), [Mac](https://en.wikipedia.org/wiki/MacOS), or [Linux](https://en.wikipedia.org/wiki/Linux)) can help diagnose the onboard system, read and erase DTCs, turn off MIL, show real-time data, and measure vehicle fuel economy.[[38]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-38)

To use OBD-II software, one needs to have an OBD-II adapter (commonly using [Bluetooth](https://en.wikipedia.org/wiki/Bluetooth), [Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi) or [USB](https://en.wikipedia.org/wiki/USB))[[39]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-39) plugged in the OBD-II port to enable the vehicle to connect with the computer where the software is installed.[[40]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-40)

**PC-based scan tools and analysis platforms**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=21)]

[](https://en.wikipedia.org/wiki/File:Obd_usb_kkl_interface.jpg)Typical simple USB KKL Diagnostic Interface without protocol logic for signal level adjustment.

A PC-based OBD analysis tool that converts the OBD-II signals to serial data (USB or serial port) standard to PCs or Macs. The software then decodes the received data to a visual display. Many popular interfaces are based on the [ELM327](https://en.wikipedia.org/wiki/ELM327) or STN[[41]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-41) OBD Interpreter ICs, both of which read all five generic OBD-II protocols. Some adapters now use the J2534 API allowing them to access OBD-II Protocols for both cars and trucks.

In addition to the functions of a hand-held scan tool, the PC-based tools generally offer:

* Large storage capacity for data logging and other functions
* Higher resolution screen than handheld tools
* The ability to use multiple software programs adding flexibility
* The identification and clearance of fault code
* Data shown by intuitive graphs and charts

The extent that a PC tool may access manufacturer or vehicle-specific ECU diagnostics varies between software products[[42]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-42) as it does between hand-held scanners.

**Data loggers**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=22)]

[](https://en.wikipedia.org/wiki/File:Texa_obd-log.png)TEXA OBD log. Small data logger with the possibility to read out the data later on PC via USB.

[Data loggers](https://en.wikipedia.org/wiki/Data_logger) are designed to capture vehicle data while the vehicle is in normal operation, for later analysis.

Data logging uses include:

* Engine and vehicle monitoring under normal operation, for diagnosis or tuning.
* Some US auto insurance companies offer reduced premiums if OBD-II vehicle data loggers[[43]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-43)[[44]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-44) or cameras[[45]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-45) are installed – and if the driver's behaviour meets requirements. This is a form of [auto insurance risk selection](https://en.wikipedia.org/wiki/Auto_insurance_risk_selection)
* Monitoring of driver behaviour by [fleet vehicle](https://en.wikipedia.org/wiki/Fleet_vehicle) operators.

Analysis of vehicle [black box](https://en.wikipedia.org/wiki/Flight_recorder) data may be performed periodically, automatically transmitted wirelessly to a third party or retrieved for forensic analysis after an event such as an accident, traffic infringement or mechanical fault.

**Emission testing**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=23)]

In the United States, many states now use OBD-II testing instead of tailpipe testing in OBD-II compliant vehicles (1996 and newer). Since OBD-II stores trouble codes for emissions equipment, the testing computer can query the vehicle's onboard computer and verify there are no emission related trouble codes and that the vehicle is in compliance with emission standards for the model year it was manufactured.

In the Netherlands, 2006 and later vehicles get a yearly EOBD emission check.[[46]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-46)

**Driver's supplementary vehicle instrumentation**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=24)]

**Driver's supplementary vehicle instrumentation** is instrumentation installed in a vehicle in addition to that provided by the vehicle manufacturer and intended for display to the driver during normal operation. This is opposed to scanners used primarily for [active fault](https://en.wikipedia.org/wiki/Active_fault) diagnosis, tuning, or hidden data logging.

Auto enthusiasts have traditionally installed additional gauges such as manifold vacuum, battery current etc. The OBD standard interface has enabled a new generation of enthusiast instrumentation accessing the full range of vehicle data used for diagnostics, and derived data such as instantaneous fuel economy.

Instrumentation may take the form of dedicated [trip computers](https://en.wikipedia.org/wiki/Trip_computer),[[47]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-47) [carputer](https://en.wikipedia.org/wiki/Carputer) or interfaces to [PDAs](https://en.wikipedia.org/wiki/Personal_digital_assistant),[[48]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-48) smartphones, or a [Garmin](https://en.wikipedia.org/wiki/Garmin) navigation unit.

As a carputer is essentially a PC, the same software could be loaded as for PC-based scan tools and vice versa, so the distinction is only in the reason for use of the software.

These enthusiast systems may also include some functionality similar to the other scan tools.

**Vehicle telematics**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=25)]

OBD II information is commonly used by vehicle telematics devices that perform fleet tracking, monitor fuel efficiency, prevent unsafe driving, as well as for remote diagnostics and by Pay-As-You-Drive insurance.

Although originally not intended for the above purposes, commonly supported OBD II data such as vehicle speed, RPM, and fuel level allow GPS-based fleet tracking devices to monitor vehicle idling times, speeding, and over-revving. By monitoring OBD II DTCs a company can know immediately if one of its vehicles has an engine problem and by interpreting the code the nature of the problem. It can be used to detect reckless driving in real time based on the sensor data provided through the OBD port.[[49]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-49) This detection is done by adding a complex events processor (CEP) to the backend and on the client's interface. OBD II is also monitored to block mobile phones when driving and to record trip data for insurance purposes.[[50]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-50)

**OBD-II diagnostic trouble codes**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=26)]

OBD-II diagnostic trouble codes ([DTCs](https://en.wikipedia.org/wiki/OBD-II_PIDs))[[51]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-obdadvisor_codes-51)[[52]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-52) are five characters long, with the first letter indicating a category, and the remaining four being a [hexadecimal](https://en.wikipedia.org/wiki/Hexadecimal) number.[[53]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-elm327dsl_pdf_p36-53)

The first character, representing category can only be one of the following four letters, given here with their associated meanings. (This restriction in number is due to how only two [bits](https://en.wikipedia.org/wiki/Bit) of memory are used to indicate the category when DTCs are stored and transmitted).[[53]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-elm327dsl_pdf_p36-53)

* P – Powertrain (engine, transmission and ignition)
* C – Chassis (includes ABS and brake fluid)
* B – Body (includes air conditioning and airbag)
* U – Network[[a]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-54) (wiring bus)
  1. [**^**](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_ref-54) Whilst this is commonly referred to as the network category, it may originally have been the 'undefined' category, hence the use of the letter 'U' rather than 'N'.

The second character is a number in the range of 0–3. (This restriction is again due to memory storage limitations).[[53]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-elm327dsl_pdf_p36-53)

* 0 – Indicates a generic (SAE defined) code.
* 1 – Indicates a manufacturer-specific (OEM) code.
* 2 – Category dependent:
  + For the 'P' category this indicates a generic (SAE defined) code.
  + For other categories indicates a manufacturer-specific (OEM) code.
* 3 – Category dependent:
  + For the 'P' category this is indicates a code that has been 'jointly' defined.
  + For other categories this has been reserved for future use.

The third character may denote a particular vehicle system that the fault relates to.[[51]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-obdadvisor_codes-51)

* 0 – Fuel and air metering and auxiliary emission controls
* 1 – Fuel and air metering
* 2 – Fuel and air metering (injector circuit)
* 3 – Ignition systems or misfires
* 4 – Auxiliary emission controls
* 5 – Vehicle speed control and idle control systems
* 6 – Computer and output circuit
* 7 – Transmission
* 8 – Transmission
* A-F – Hybrid Trouble Codes

Finally the fourth and fifth characters define the exact problem detected.

**Standards documents**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=27)]

**SAE standards documents on OBD-II**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=28)]

* J1962 – Defines the physical connector used for the OBD-II interface.
* J1850 – Defines a serial data protocol. There are 2 variants: 10.4 kbit/s (single wire, VPW) and 41.6 kbit/s (2 wire, PWM). Mainly used by US manufacturers, also known as PCI (Chrysler, 10.4K), Class 2 (GM, 10.4K), and SCP (Ford, 41.6K)
* J1978 – Defines minimal operating standards for OBD-II scan tools
* J1979 – Defines standards for diagnostic test modes
* J2012 – Defines standards trouble codes and definitions.
* J2178-1 – Defines standards for network message header formats and physical address assignments
* J2178-2 – Gives data parameter definitions
* J2178-3 – Defines standards for network message frame IDs for single byte headers
* J2178-4 – Defines standards for network messages with three byte headers\*
* J2284-3 – Defines 500K [CAN](https://en.wikipedia.org/wiki/Controller_area_network) [physical](https://en.wikipedia.org/wiki/Physical_layer) and [data link layer](https://en.wikipedia.org/wiki/Data_link_layer)
* J2411 – Describes the [GMLAN](https://en.wikipedia.org/wiki/General_Motors_Local_Area_Network) (Single-Wire CAN) protocol, used in newer GM vehicles. Often accessible on the OBD connector as PIN 1 on newer GM vehicles.

**SAE standards documents on HD (Heavy Duty) OBD**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=29)]

* [J1939](https://en.wikipedia.org/wiki/J1939) – Defines a data protocol for heavy duty commercial vehicles

**ISO standards**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=30)]

* ISO 9141: Road vehicles – Diagnostic systems. [International Organization for Standardization](https://en.wikipedia.org/wiki/International_Organization_for_Standardization), 1989.
  + Part 1: Requirements for interchange of digital information
  + Part 2: CARB requirements for interchange of digital information
  + Part 3: Verification of the communication between vehicle and OBD II scan tool
* ISO 11898: Road vehicles – Controller area network (CAN). International Organization for Standardization, 2003.
  + Part 1: Data link layer and physical signalling
  + Part 2: High-speed medium access unit
  + Part 3: Low-speed, fault-tolerant, medium-dependent interface
  + Part 4: Time-triggered communication
* ISO 14230: Road vehicles – Diagnostic systems – Keyword Protocol 2000, International Organization for Standardization, 1999.
  + Part 1: Physical layer
  + Part 2: Data link layer
  + Part 3: Application layer
  + Part 4: Requirements for emission-related systems
* ISO 15031: Communication between vehicle and external equipment for emissions-related diagnostics, International Organization for Standardization, 2010.
  + Part 1: General information and use case definition
  + Part 2: Guidance on terms, definitions, abbreviations and acronyms
  + Part 3: Diagnostic connector and related electrical circuits, specification and use
  + Part 4: External test equipment
  + Part 5: Emissions-related diagnostic services
  + Part 6: Diagnostic trouble code definitions
  + Part 7: Data link security
* ISO 15765: Road vehicles – Diagnostics on Controller Area Networks (CAN). International Organization for Standardization, 2004.
  + Part 1: General information
  + Part 2: Network layer services [ISO 15765-2](https://en.wikipedia.org/wiki/ISO_15765-2)
  + Part 3: Implementation of unified diagnostic services ([UDS](https://en.wikipedia.org/wiki/Unified_Diagnostic_Services) on CAN)
  + Part 4: Requirements for emissions-related systems

**Security issues**

[[edit](https://en.wikipedia.org/w/index.php?title=On-board_diagnostics&action=edit&section=31)]

Researchers at the [University of Washington](https://en.wikipedia.org/wiki/University_of_Washington) and [University of California](https://en.wikipedia.org/wiki/University_of_California) examined the security around OBD and found that they were able to gain control over many vehicle components via the interface. Furthermore, they were able to upload new [firmware](https://en.wikipedia.org/wiki/Firmware) into the [engine control units](https://en.wikipedia.org/wiki/Engine_control_unit). Their conclusion is that vehicle [embedded systems](https://en.wikipedia.org/wiki/Embedded_system) are not designed with security in mind.[[54]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-55)[[55]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-56)[[56]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-57)

There have been reports of thieves using specialist OBD reprogramming devices to enable them to steal cars without the use of a key.[[57]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-58) The primary causes of this vulnerability lie in the tendency for vehicle manufacturers to extend the [bus](https://en.wikipedia.org/wiki/Bus_(computing)) for purposes other than those for which it was designed, and the lack of [authentication](https://en.wikipedia.org/wiki/Authentication) and [authorization](https://en.wikipedia.org/wiki/Authorization) in the OBD specifications, which instead rely largely on [security through obscurity](https://en.wikipedia.org/wiki/Security_through_obscurity).[[58]](https://en.wikipedia.org/wiki/On-board_diagnostics#cite_note-59)

From Wikipedia, the free encyclopedia

|  |  |  |  |
| --- | --- | --- | --- |
| https://upload.wikimedia.org/wikipedia/en/thumb/b/b4/Ambox_important.svg/40px-Ambox_important.svg.png | hide**This article has multiple issues.** Please help [**improve it**](https://en.wikipedia.org/wiki/Special:EditPage/OBD-II_PIDs) or discuss these issues on the [**talk page**](https://en.wikipedia.org/wiki/Talk:OBD-II_PIDs). *(*[*Learn how and when to remove these messages*](https://en.wikipedia.org/wiki/Help:Maintenance_template_removal)*)*   |  | | --- | | This article **may be too technical for most readers to understand**. *(December 2023)* |  |  | | --- | | This article **is written like**[**a manual or guide**](https://en.wikipedia.org/wiki/Wikipedia:What_Wikipedia_is_not#GUIDE)**.** *(December 2023)* | |

**OBD-II PIDs** ([On-board diagnostics](https://en.wikipedia.org/wiki/On-board_diagnostics) **Parameter IDs**) are codes used to request data from a vehicle, used as a diagnostic tool.

[SAE](https://en.wikipedia.org/wiki/SAE_International) standard J1979 defines many OBD-II PIDs. All on-road vehicles and trucks sold in North America are required to support a subset of these codes, primarily for state mandated [emissions](https://en.wikipedia.org/wiki/Vehicle_emissions_control) [inspections](https://en.wikipedia.org/wiki/Vehicle_inspection). Manufacturers also define additional PIDs specific to their vehicles. Though not mandated, many motorcycles also support OBD-II PIDs.

In 1996, light duty vehicles (less than 8,500 lb or 3,900 kg) were the first to be mandated followed by medium duty vehicles (8,500–14,000 lb or 3,900–6,400 kg) in 2005.[[1]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-epa-1) They are both required to be accessed through a standardized [data link connector](https://en.wikipedia.org/wiki/Data_link_connector_(automotive)) defined by [SAE J1962](https://en.wikipedia.org/wiki/SAE_J1962).

Heavy duty vehicles (greater than 14,000 lb or 6,400 kg) made after 2010,[[1]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-epa-1) for sale in the US are allowed to support OBD-II diagnostics through [SAE standard J1939-13](https://en.wikipedia.org/wiki/SAE_J1939) (a round diagnostic connector) according to CARB in title 13 CCR 1971.1. Some heavy duty trucks in North America use the SAE J1962 OBD-II diagnostic connector that is common with passenger cars, notably Mack and Volvo Trucks, however they use 29 bit CAN identifiers (unlike 11 bit headers used by passenger cars).

**Services / Modes**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=1)]

There are 10 diagnostic services described in the latest OBD-II standard SAE J1979. Before 2002, J1979 referred to these services as "modes". They are as follows:

|  |  |
| --- | --- |
| **Service / Mode (hex)** | **Description** |
| [01](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_01) | Show current data |
| [02](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_02) | Show freeze frame data |
| [03](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_03) | Show stored Diagnostic Trouble Codes |
| [04](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_04) | Clear Diagnostic Trouble Codes and stored values |
| [05](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_05) | Test results, oxygen sensor monitoring (non CAN only) |
| 06 | Test results, other component/system monitoring (Test results, oxygen sensor monitoring for CAN only) |
| 07 | Show pending Diagnostic Trouble Codes (detected during current or last driving cycle) |
| 08 | Control operation of on-board component/system |
| [09](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_09) | Request vehicle information |
| 0A | Permanent [Diagnostic Trouble Codes](https://en.wikipedia.org/wiki/Diagnostic_Trouble_Code) (DTCs) (Cleared DTCs) |

Vehicle manufacturers are not required to support all services. Each manufacturer may define additional services above #9 (e.g.: service 22 as defined by SAE J2190 for Ford/GM, service 21 for Toyota) for other information e.g. the voltage of the traction battery in a [hybrid electric vehicle](https://en.wikipedia.org/wiki/Hybrid_electric_vehicle) (HEV).[[2]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-2)

The nonOBD [UDS](https://en.wikipedia.org/wiki/Unified_Diagnostic_Services) services start at 0x10 to avoid overlap of ID-range.

**Standard PIDs**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=2)]

The table below shows the standard OBD-II PIDs as defined by SAE J1979. The expected response for each PID is given, along with information on how to translate the response into meaningful data. Again, not all vehicles will support all PIDs and there can be manufacturer-defined custom PIDs that are not defined in the OBD-II standard.

Note that services 01 and 02 are basically identical, except that service 01 provides current information, whereas service 02 provides a snapshot of the same data taken at the point when the last diagnostic trouble code was set. The exceptions are PID 01, which is only available in service 01, and PID 02, which is only available in service 02. If service 02 PID 02 returns zero, then there is no snapshot and all other service 02 data is meaningless.

When using Bit-Encoded-Notation, quantities like C4 means bit 4 from data byte C. Each bit is numbered from 0 to 7, so 7 is the most significant bit and 0 is the least significant bit ([See below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_01_PID_00)).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | | | | | | | | B | | | | | | | | C | | | | | | | | D | | | | | | | |
| A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 | C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

**Service 01 - Show current data**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=3)]

| **PIDs (hex)** | **PID (Dec)** | **Data bytes returned** | **Description** | **Min value** | **Max value** | **Units** | **Formula**[[a]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-formula-3) |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 00 | 0 | 4 | PIDs supported [$01 - $20] |  |  |  | Bit encoded [A7..D0] == [PID $01..PID $20] [See below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_01_PID_00) |
| 01 | 1 | 4 | Monitor status since DTCs cleared. (Includes malfunction indicator lamp (MIL), status and number of DTCs, components tests, DTC readiness checks) |  |  |  | Bit encoded. [See below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_01_PID_01) |
| 02 | 2 | 2 | DTC that caused freeze frame to be stored. |  |  |  | [Decoded as in service 3](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_03_(no_PID_required)) |
| 03 | 3 | 2 | Fuel system status |  |  |  | Bit encoded. [See below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_01_PID_03) |
| 04 | 4 | 1 | Calculated engine load | 0 | 100 | % | 100255A (or A2.55) |
| 05 | 5 | 1 | Engine coolant temperature | -40 | 215 | °C | A−40 |
| 06 | 6 | 1 | Short term fuel trim (STFT)—Bank 1 | -100 (Reduce Fuel: Too Rich) | 99.2 (Add Fuel: Too Lean) | % | 100128A−100(or A1.28−100 ) |
| 07 | 7 | 1 | Long term fuel trim (LTFT)—Bank 1 |
| 08 | 8 | 1 | Short term fuel trim (STFT)—Bank 2 |
| 09 | 9 | 1 | Long term fuel trim (LTFT)—Bank 2 |
| 0A | 10 | 1 | Fuel pressure ([gauge pressure](https://en.wikipedia.org/wiki/Pressure_measurement#Absolute,_gauge_and_differential_pressures_-_zero_reference)) | 0 | 765 | kPa | 3A |
| 0B | 11 | 1 | Intake manifold absolute pressure | 0 | 255 | kPa | A |
| 0C | 12 | 2 | Engine speed | 0 | 16,383.75 | rpm | 256A+B4 |
| 0D | 13 | 1 | Vehicle speed | 0 | 255 | km/h | A |
| 0E | 14 | 1 | Timing advance | -64 | 63.5 | ° before [TDC](https://en.wikipedia.org/wiki/Dead_centre_(engineering)) | A2−64 |
| 0F | 15 | 1 | Intake air temperature | -40 | 215 | °C | A−40 |
| 10 | 16 | 2 | [Mass air flow sensor (MAF)](https://en.wikipedia.org/wiki/Mass_airflow_sensor) air flow rate | 0 | 655.35 | g/s | 256A+B100 |
| 11 | 17 | 1 | Throttle position | 0 | 100 | % | 100255A |
| 12 | 18 | 1 | Commanded secondary air status |  |  |  | Bit encoded. [See below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_01_PID_12) |
| 13 | 19 | 1 | Oxygen sensors present (in 2 banks) |  |  |  | [A0..A3] == Bank 1, Sensors 1-4. [A4..A7] == Bank 2... |
| 14 | 20 | 2 | Oxygen Sensor 1 A: Voltage B: Short term fuel trim | 0 -100 | 1.275 99.2 | V  % | A200100128B−100(if B==$FF, sensor is not used in trim calculation) |
| 15 | 21 | 2 | Oxygen Sensor 2 A: Voltage B: Short term fuel trim |
| 16 | 22 | 2 | Oxygen Sensor 3 A: Voltage B: Short term fuel trim |
| 17 | 23 | 2 | Oxygen Sensor 4 A: Voltage B: Short term fuel trim |
| 18 | 24 | 2 | Oxygen Sensor 5 A: Voltage B: Short term fuel trim |
| 19 | 25 | 2 | Oxygen Sensor 6 A: Voltage B: Short term fuel trim |
| 1A | 26 | 2 | Oxygen Sensor 7 A: Voltage B: Short term fuel trim |
| 1B | 27 | 2 | Oxygen Sensor 8 A: Voltage B: Short term fuel trim |
| 1C | 28 | 1 | OBD standards this vehicle conforms to | 1 | 250 |  | [enumerated](https://en.wikipedia.org/wiki/Enumerated_type). [See below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_01_PID_1C) |
| 1D | 29 | 1 | Oxygen sensors present (in 4 banks) |  |  |  | Similar to PID $13, but [A0..A7] == [B1S1, B1S2, B2S1, B2S2, B3S1, B3S2, B4S1, B4S2] |
| 1E | 30 | 1 | Auxiliary input status |  |  |  | A0 == [Power Take Off](https://en.wikipedia.org/wiki/Power_Take_Off) (PTO) status (1 == active) [A1..A7] not used |
| 1F | 31 | 2 | Run time since engine start | 0 | 65,535 | s | 256A+B |
| 20 | 32 | 4 | PIDs supported [$21 - $40] |  |  |  | Bit encoded [A7..D0] == [PID $21..PID $40] [See below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_01_PID_00) |
| 21 | 33 | 2 | Distance traveled with malfunction indicator lamp (MIL) on | 0 | 65,535 | km | 256A+B |
| 22 | 34 | 2 | [Fuel Rail](https://en.wikipedia.org/wiki/Fuel_rail) Pressure (relative to manifold vacuum) | 0 | 5177.265 | kPa | 0.079(256A+B) |
| 23 | 35 | 2 | [Fuel Rail](https://en.wikipedia.org/wiki/Fuel_rail) Gauge Pressure (diesel, or gasoline direct injection) | 0 | 655,350 | kPa | 10(256A+B) |
| 24 | 36 | 4 | Oxygen Sensor 1 AB: Air-Fuel Equivalence Ratio ([lambda,λ](https://en.wikipedia.org/wiki/Air%E2%80%93fuel_ratio#Air%E2%80%93fuel_equivalence_ratio_(%CE%BB))) CD: Voltage | 0 0 | < 2 < 8 | ratio V | 265536(256A+B)865536(256C+D) |
| 25 | 37 | 4 | Oxygen Sensor 2 AB: Air-Fuel Equivalence Ratio ([lambda,λ](https://en.wikipedia.org/wiki/Air%E2%80%93fuel_ratio#Air%E2%80%93fuel_equivalence_ratio_(%CE%BB))) CD: Voltage |
| 26 | 38 | 4 | Oxygen Sensor 3 AB: Air-Fuel Equivalence Ratio ([lambda,λ](https://en.wikipedia.org/wiki/Air%E2%80%93fuel_ratio#Air%E2%80%93fuel_equivalence_ratio_(%CE%BB))) CD: Voltage |
| 27 | 39 | 4 | Oxygen Sensor 4 AB: Air-Fuel Equivalence Ratio ([lambda,λ](https://en.wikipedia.org/wiki/Air%E2%80%93fuel_ratio#Air%E2%80%93fuel_equivalence_ratio_(%CE%BB))) CD: Voltage |
| 28 | 40 | 4 | Oxygen Sensor 5 AB: Air-Fuel Equivalence Ratio ([lambda,λ](https://en.wikipedia.org/wiki/Air%E2%80%93fuel_ratio#Air%E2%80%93fuel_equivalence_ratio_(%CE%BB))) CD: Voltage |
| 29 | 41 | 4 | Oxygen Sensor 6 AB: Air-Fuel Equivalence Ratio ([lambda,λ](https://en.wikipedia.org/wiki/Air%E2%80%93fuel_ratio#Air%E2%80%93fuel_equivalence_ratio_(%CE%BB))) CD: Voltage |
| 2A | 42 | 4 | Oxygen Sensor 7 AB: Air-Fuel Equivalence Ratio ([lambda,λ](https://en.wikipedia.org/wiki/Air%E2%80%93fuel_ratio#Air%E2%80%93fuel_equivalence_ratio_(%CE%BB))) CD: Voltage |
| 2B | 43 | 4 | Oxygen Sensor 8 AB: Air-Fuel Equivalence Ratio ([lambda,λ](https://en.wikipedia.org/wiki/Air%E2%80%93fuel_ratio#Air%E2%80%93fuel_equivalence_ratio_(%CE%BB))) CD: Voltage |
| 2C | 44 | 1 | Commanded [EGR](https://en.wikipedia.org/wiki/Exhaust_gas_recirculation) | 0 | 100 | % | 100255A |
| 2D | 45 | 1 | EGR Error | -100 | 99.2 | % | 100128A−100 |
| 2E | 46 | 1 | Commanded evaporative purge | 0 | 100 | % | 100255A |
| 2F | 47 | 1 | Fuel Tank Level Input | 0 | 100 | % | 100255A |
| 30 | 48 | 1 | Warm-ups since codes cleared | 0 | 255 |  | A |
| 31 | 49 | 2 | Distance traveled since codes cleared | 0 | 65,535 | km | 256A+B |
| 32 | 50 | 2 | Evap. System Vapor Pressure | -8,192 | 8191.75 | Pa | 256A+B4  (AB is [two's complement](https://en.wikipedia.org/wiki/Two%27s_complement) signed)[[3]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-:0-4) |
| 33 | 51 | 1 | Absolute Barometric Pressure | 0 | 255 | kPa | A |
| 34 | 52 | 4 | Oxygen Sensor 1 AB: Air-Fuel Equivalence Ratio ([lambda,λ](https://en.wikipedia.org/wiki/Air%E2%80%93fuel_ratio#Air%E2%80%93fuel_equivalence_ratio_(%CE%BB))) CD: Current | 0 -128 | < 2 <128 | ratio mA | 265536(256A+B)256C+D256−128 |
| 35 | 53 | 4 | Oxygen Sensor 2 AB: Air-Fuel Equivalence Ratio ([lambda,λ](https://en.wikipedia.org/wiki/Air%E2%80%93fuel_ratio#Air%E2%80%93fuel_equivalence_ratio_(%CE%BB))) CD: Current |
| 36 | 54 | 4 | Oxygen Sensor 3 AB: Air-Fuel Equivalence Ratio ([lambda,λ](https://en.wikipedia.org/wiki/Air%E2%80%93fuel_ratio#Air%E2%80%93fuel_equivalence_ratio_(%CE%BB))) CD: Current |
| 37 | 55 | 4 | Oxygen Sensor 4 AB: Air-Fuel Equivalence Ratio ([lambda,λ](https://en.wikipedia.org/wiki/Air%E2%80%93fuel_ratio#Air%E2%80%93fuel_equivalence_ratio_(%CE%BB))) CD: Current |
| 38 | 56 | 4 | Oxygen Sensor 5 AB: Air-Fuel Equivalence Ratio ([lambda,λ](https://en.wikipedia.org/wiki/Air%E2%80%93fuel_ratio#Air%E2%80%93fuel_equivalence_ratio_(%CE%BB))) CD: Current |
| 39 | 57 | 4 | Oxygen Sensor 6 AB: Air-Fuel Equivalence Ratio ([lambda,λ](https://en.wikipedia.org/wiki/Air%E2%80%93fuel_ratio#Air%E2%80%93fuel_equivalence_ratio_(%CE%BB))) CD: Current |
| 3A | 58 | 4 | Oxygen Sensor 7 AB: Air-Fuel Equivalence Ratio ([lambda,λ](https://en.wikipedia.org/wiki/Air%E2%80%93fuel_ratio#Air%E2%80%93fuel_equivalence_ratio_(%CE%BB))) CD: Current |
| 3B | 59 | 4 | Oxygen Sensor 8 AB: Air-Fuel Equivalence Ratio ([lambda,λ](https://en.wikipedia.org/wiki/Air%E2%80%93fuel_ratio#Air%E2%80%93fuel_equivalence_ratio_(%CE%BB))) CD: Current |
| 3C | 60 | 2 | Catalyst Temperature: Bank 1, Sensor 1 | -40 | 6,513.5 | °C | 256A+B10−40 |
| 3D | 61 | 2 | Catalyst Temperature: Bank 2, Sensor 1 |
| 3E | 62 | 2 | Catalyst Temperature: Bank 1, Sensor 2 |
| 3F | 63 | 2 | Catalyst Temperature: Bank 2, Sensor 2 |
| 40 | 64 | 4 | PIDs supported [$41 - $60] |  |  |  | Bit encoded [A7..D0] == [PID $41..PID $60] [See below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_01_PID_00) |
| 41 | 65 | 4 | Monitor status this drive cycle |  |  |  | Bit encoded. [See below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_01_PID_41) |
| 42 | 66 | 2 | Control module voltage | 0 | 65.535 | V | 256A+B1000 |
| 43 | 67 | 2 | Absolute load value | 0 | 25,700 | % | 100255(256A+B) |
| 44 | 68 | 2 | Commanded Air-Fuel Equivalence Ratio ([lambda,λ](https://en.wikipedia.org/wiki/Air%E2%80%93fuel_ratio#Air%E2%80%93fuel_equivalence_ratio_(%CE%BB))) | 0 | < 2 | ratio | 265536(256A+B) |
| 45 | 69 | 1 | Relative throttle position | 0 | 100 | % | 100255A |
| 46 | 70 | 1 | Ambient air temperature | -40 | 215 | °C | A−40 |
| 47 | 71 | 1 | Absolute throttle position B | 0 | 100 | % | 100255A |
| 48 | 72 | 1 | Absolute throttle position C |
| 49 | 73 | 1 | Accelerator pedal position D |
| 4A | 74 | 1 | Accelerator pedal position E |
| 4B | 75 | 1 | Accelerator pedal position F |
| 4C | 76 | 1 | Commanded throttle actuator |
| 4D | 77 | 2 | Time run with MIL on | 0 | 65,535 | min | 256A+B |
| 4E | 78 | 2 | Time since trouble codes cleared |
| 4F | 79 | 4 | Maximum value for Fuel–Air equivalence ratio, oxygen sensor voltage, oxygen sensor current, and intake manifold absolute pressure | 0, 0, 0, 0 | 255, 255, 255, 2550 | ratio, V, mA, kPa | A, B, C, D×10 |
| 50 | 80 | 4 | Maximum value for air flow rate from mass air flow sensor | 0 | 2550 | g/s | A×10; B, C, and D are reserved for future use |
| 51 | 81 | 1 | Fuel Type |  |  |  | From fuel type table [see below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Fuel_Type_Coding) |
| 52 | 82 | 1 | Ethanol fuel % | 0 | 100 | % | 100255A |
| 53 | 83 | 2 | Absolute Evap system Vapor Pressure | 0 | 327.675 | kPa | 256A+B200 |
| 54 | 84 | 2 | Evap system vapor pressure | -32,768 | 32,767 | Pa | 256A+B(AB is [two's complement](https://en.wikipedia.org/wiki/Two%27s_complement) signed)[[3]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-:0-4) |
| 55 | 85 | 2 | Short term secondary oxygen sensor trim, A: bank 1, B: bank 3 | -100 | 99.2 | % | 100128A−100  100128B−100 |
| 56 | 86 | 2 | Long term secondary oxygen sensor trim, A: bank 1, B: bank 3 |
| 57 | 87 | 2 | Short term secondary oxygen sensor trim, A: bank 2, B: bank 4 |
| 58 | 88 | 2 | Long term secondary oxygen sensor trim, A: bank 2, B: bank 4 |
| 59 | 89 | 2 | [Fuel rail](https://en.wikipedia.org/wiki/Fuel_rail) absolute pressure | 0 | 655,350 | kPa | 10(256A+B) |
| 5A | 90 | 1 | Relative accelerator pedal position | 0 | 100 | % | 100255A |
| 5B | 91 | 1 | Hybrid battery pack remaining life | 0 | 100 | % | 100255A |
| 5C | 92 | 1 | Engine oil temperature | -40 | 210 | °C | A−40 |
| 5D | 93 | 2 | Fuel injection timing | -210.00 | 301.992 | ° | 256A+B128−210 |
| 5E | 94 | 2 | Engine fuel rate | 0 | 3212.75 | L/h | 256A+B20 |
| 5F | 95 | 1 | Emission requirements to which vehicle is designed |  |  |  | Bit Encoded |
| 60 | 96 | 4 | PIDs supported [$61 - $80] |  |  |  | Bit encoded [A7..D0] == [PID $61..PID $80] [See below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_01_PID_00) |
| 61 | 97 | 1 | Driver's demand engine - percent torque | -125 | 130 | % | A−125 |
| 62 | 98 | 1 | Actual engine - percent torque | -125 | 130 | % | A−125 |
| 63 | 99 | 2 | Engine reference torque | 0 | 65,535 | N⋅m | 256A+B |
| 64 | 100 | 5 | Engine percent torque data | -125 | 130 | % | A−125 Idle B−125 Engine point 1 C−125 Engine point 2 D−125 Engine point 3 E−125 Engine point 4 |
| 65 | 101 | 2 | Auxiliary input / output supported |  |  |  | Bit Encoded |
| 66 | 102 | 5 | Mass air flow sensor | 0 | 2047.96875 | g/s | [A0]== Sensor A Supported [A1]== Sensor B Supported Sensor A:256B+C32 Sensor B:256D+E32 |
| 67 | 103 | 3 | Engine coolant temperature | -40 | 215 | °C | [A0]== Sensor 1 Supported [A1]== Sensor 2 Supported Sensor 1:B−40 Sensor 2:C−40 |
| 68 | 104 | 3 | Intake air temperature sensor | -40 | 215 | °C | [A0]== Sensor 1 Supported [A1]== Sensor 2 Supported Sensor 1:B−40 Sensor 2:C−40 |
| 69 | 105 | 7 | Actual EGR, Commanded EGR, and EGR Error |  |  |  |  |
| 6A | 106 | 5 | Commanded Diesel intake air flow control and relative intake air flow position |  |  |  |  |
| 6B | 107 | 5 | Exhaust gas recirculation temperature |  |  |  |  |
| 6C | 108 | 5 | Commanded throttle actuator control and relative throttle position |  |  |  |  |
| 6D | 109 | 11 | Fuel pressure control system |  |  |  |  |
| 6E | 110 | 9 | Injection pressure control system |  |  |  |  |
| 6F | 111 | 3 | Turbocharger compressor inlet pressure |  |  |  |  |
| 70 | 112 | 10 | Boost pressure control |  |  |  |  |
| 71 | 113 | 6 | Variable Geometry turbo (VGT) control |  |  |  |  |
| 72 | 114 | 5 | Wastegate control |  |  |  |  |
| 73 | 115 | 5 | Exhaust pressure |  |  |  |  |
| 74 | 116 | 5 | Turbocharger RPM |  |  |  |  |
| 75 | 117 | 7 | Turbocharger temperature |  |  |  |  |
| 76 | 118 | 7 | Turbocharger temperature |  |  |  |  |
| 77 | 119 | 5 | Charge air cooler temperature (CACT) |  |  |  |  |
| 78 | 120 | 9 | Exhaust Gas temperature (EGT) Bank 1 |  |  |  | Special PID. [See below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_01_PID_78) |
| 79 | 121 | 9 | Exhaust Gas temperature (EGT) Bank 2 |  |  |  | Special PID. [See below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_01_PID_78) |
| 7A | 122 | 7 | Diesel particulate filter (DPF)  differential pressure |  |  |  |  |
| 7B | 123 | 7 | Diesel particulate filter (DPF) |  |  |  |  |
| 7C | 124 | 9 | Diesel Particulate filter (DPF) temperature |  |  | °C | 256A+B10−40 |
| 7D | 125 | 1 | NOx NTE ([Not-To-Exceed](https://en.wikipedia.org/wiki/Not-To-Exceed)) control area status |  |  |  |  |
| 7E | 126 | 1 | PM NTE ([Not-To-Exceed](https://en.wikipedia.org/wiki/Not-To-Exceed)) control area status |  |  |  |  |
| 7F | 127 | 13 | Engine run time [[b]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-6) |  |  | s |  |
| 80 | 128 | 4 | PIDs supported [$81 - $A0] |  |  |  | Bit encoded [A7..D0] == [PID $81..PID $A0] [See below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_01_PID_00) |
| 81 | 129 | 41 | Engine run time for Auxiliary Emissions Control Device(AECD) |  |  |  |  |
| 82 | 130 | 41 | Engine run time for Auxiliary Emissions Control Device(AECD) |  |  |  |  |
| 83 | 131 | 9 | NOx sensor |  |  |  |  |
| 84 | 132 | 1 | Manifold surface temperature |  |  |  |  |
| 85 | 133 | 10 | NOx reagent system |  |  | % | 100255E |
| 86 | 134 | 5 | Particulate matter (PM) sensor |  |  |  |  |
| 87 | 135 | 5 | Intake manifold absolute pressure |  |  |  |  |
| 88 | 136 | 13 | SCR Induce System |  |  |  |  |
| 89 | 137 | 41 | Run Time for AECD #11-#15 |  |  |  |  |
| 8A | 138 | 41 | Run Time for AECD #16-#20 |  |  |  |  |
| 8B | 139 | 7 | Diesel Aftertreatment |  |  |  |  |
| 8C | 140 | 17 | O2 Sensor (Wide Range) |  |  |  |  |
| 8D | 141 | 1 | Throttle Position G | 0 | 100 | % |  |
| 8E | 142 | 1 | Engine Friction - Percent Torque | -125 | 130 | % | A−125 |
| 8F | 143 | 7 | PM Sensor Bank 1 & 2 |  |  |  |  |
| 90 | 144 | 3 | WWH-OBD Vehicle OBD System Information |  |  | h |  |
| 91 | 145 | 5 | WWH-OBD Vehicle OBD System Information |  |  | h |  |
| 92 | 146 | 2 | Fuel System Control |  |  |  |  |
| 93 | 147 | 3 | WWH-OBD Vehicle OBD Counters support |  |  | h |  |
| 94 | 148 | 12 | NOx Warning And Inducement System |  |  |  |  |
| 98 | 152 | 9 | Exhaust Gas Temperature Sensor |  |  |  |  |
| 99 | 153 | 9 | Exhaust Gas Temperature Sensor |  |  |  |  |
| 9A | 154 | 6 | Hybrid/EV Vehicle System Data, Battery, Voltage |  |  |  |  |
| 9B | 155 | 4 | Diesel Exhaust Fluid Sensor Data |  |  | % | 100255D |
| 9C | 156 | 17 | O2 Sensor Data |  |  |  |  |
| 9D | 157 | 4 | Engine Fuel Rate |  |  | g/s |  |
| 9E | 158 | 2 | Engine Exhaust Flow Rate |  |  | kg/h |  |
| 9F | 159 | 9 | Fuel System Percentage Use |  |  |  |  |
| A0 | 160 | 4 | PIDs supported [$A1 - $C0] |  |  |  | Bit encoded [A7..D0] == [PID $A1..PID $C0] [See below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_01_PID_00) |
| A1 | 161 | 9 | NOx Sensor Corrected Data |  |  | ppm |  |
| A2 | 162 | 2 | Cylinder Fuel Rate | 0 | 2047.96875 | mg/stroke | 256A+B32 |
| A3 | 163 | 9 | Evap System Vapor Pressure |  |  | Pa |  |
| A4 | 164 | 4 | Transmission Actual Gear | 0 | 65.535 | ratio | [A1]==Supported  256C+D1000 |
| A5 | 165 | 4 | Commanded Diesel Exhaust Fluid Dosing | 0 | 127.5 | % | [A0]= 1:Supported; 0:Unsupported  B2 |
| A6 | 166 | 4 | [Odometer](https://en.wikipedia.org/wiki/Odometer) [[c]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-7) | 0 | 429,496,729.5 | [km](https://en.wikipedia.org/wiki/Kilometre) | A(224)+B(216)+C(28)+D10 |
| A7 | 167 | 4 | NOx Sensor Concentration Sensors 3 and 4 |  |  |  |  |
| A8 | 168 | 4 | NOx Sensor Corrected Concentration Sensors 3 and 4 |  |  |  |  |
| A9 | 169 | 4 | ABS Disable Switch State |  |  |  | [A0]= 1:Supported; 0:Unsupported  [B0]= 1:Yes;0:No |
| C0 | 192 | 4 | PIDs supported [$C1 - $E0] |  |  |  | Bit encoded [A7..D0] == [PID $C1..PID $E0] [See below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_01_PID_00) |
| C3 | 195 | 2 | Fuel Level Input A/B | 0 | 25,700 | % | Returns numerous data, including Drive Condition ID and Engine Speed\* |
| C4 | 196 | 8 | Exhaust Particulate Control System Diagnostic Time/Count | 0 | 4,294,967,295 | seconds / Count | B5 is Engine Idle Request B6 is Engine Stop Request\* First byte = Time in seconds Second byte = Count |
| C5 | 197 | 4 | Fuel Pressure A and B | 0 | 5,177 | kPa |  |
| C6 | 198 | 7 | Byte 1 - Particulate control - driver inducement system status Byte 2,3 - Removal or block of the particulate aftertreatment system counter Byte 4,5 - Liquid regent injection system (e.g. fuel-borne catalyst) failure counter Byte 6,7 - Malfunction of Particulate control monitoring system counter | 0 | 65,535 | h |  |
| C7 | 199 | 2 | Distance Since Reflash or Module Replacement | 0 | 65,535 | km |  |
| C8 | 200 | 1 | NOx Control Diagnostic (NCD) and Particulate Control Diagnostic (PCD) Warning Lamp status | - | - | Bit |  |
| **PID (hex)** | **PID (Dec)** | **Data bytes returned** | **Description** | **Min value** | **Max value** | **Units** | **Formula**[[a]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-formula-3) |

**Service 02 - Show freeze frame data**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=4)]

Service 02 accepts the same PIDs as service 01, with the same meaning,[[5]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-8) but information given is from when the freeze frame[[6]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-9) was created. Note that PID $02 is used to obtain the DTC that triggered the freeze frame.

A person has to send the frame number in the data section of the message.

**Service 03 - Show stored Diagnostic Trouble Codes (DTCs)**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=5)]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PID (hex)** | **Data bytes returned** | **Description** | **Min value** | **Max value** | **Units** | **Formula**[[a]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-formula-3) |
| N/A | n\*6 | Request trouble codes |  |  |  | 3 codes per message frame. [See below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_03_(no_PID_required)) |

**Service 04 - Clear Diagnostic Trouble Codes and stored values**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=6)]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PID (hex)** | **Data bytes returned** | **Description** | **Min value** | **Max value** | **Units** | **Formula**[[a]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-formula-3) |
| N/A | 0 | Clear trouble codes / Malfunction indicator lamp (MIL) / Check engine light |  |  |  | Clears all stored trouble codes and turns the MIL off. |

**Service 05 - Test results, oxygen sensor monitoring (non CAN only)**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=7)]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PID (hex)** | **Data bytes returned** | **Description** | **Min value** | **Max value** | **Units** | **Formula**[[a]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-formula-3) |
| 0100 | 4 | OBD Monitor IDs supported ($01 – $20) | 0x0 | 0xffffffff |  |  |
| 0101 | 2 | O2 Sensor Monitor Bank 1 Sensor 1 | 0.00 | 1.275 | V | 0.005 Rich to lean sensor threshold voltage |
| 0102 |  | O2 Sensor Monitor Bank 1 Sensor 2 | 0.00 | 1.275 | V | 0.005 Rich to lean sensor threshold voltage |
| 0103 |  | O2 Sensor Monitor Bank 1 Sensor 3 | 0.00 | 1.275 | V | 0.005 Rich to lean sensor threshold voltage |
| 0104 |  | O2 Sensor Monitor Bank 1 Sensor 4 | 0.00 | 1.275 | V | 0.005 Rich to lean sensor threshold voltage |
| 0105 |  | O2 Sensor Monitor Bank 2 Sensor 1 | 0.00 | 1.275 | V | 0.005 Rich to lean sensor threshold voltage |
| 0106 |  | O2 Sensor Monitor Bank 2 Sensor 2 | 0.00 | 1.275 | V | 0.005 Rich to lean sensor threshold voltage |
| 0107 |  | O2 Sensor Monitor Bank 2 Sensor 3 | 0.00 | 1.275 | V | 0.005 Rich to lean sensor threshold voltage |
| 0108 |  | O2 Sensor Monitor Bank 2 Sensor 4 | 0.00 | 1.275 | V | 0.005 Rich to lean sensor threshold voltage |
| 0109 |  | O2 Sensor Monitor Bank 3 Sensor 1 | 0.00 | 1.275 | V | 0.005 Rich to lean sensor threshold voltage |
| 010A |  | O2 Sensor Monitor Bank 3 Sensor 2 | 0.00 | 1.275 | V | 0.005 Rich to lean sensor threshold voltage |
| 010B |  | O2 Sensor Monitor Bank 3 Sensor 3 | 0.00 | 1.275 | V | 0.005 Rich to lean sensor threshold voltage |
| 010C |  | O2 Sensor Monitor Bank 3 Sensor 4 | 0.00 | 1.275 | V | 0.005 Rich to lean sensor threshold voltage |
| 010D |  | O2 Sensor Monitor Bank 4 Sensor 1 | 0.00 | 1.275 | V | 0.005 Rich to lean sensor threshold voltage |
| 010E |  | O2 Sensor Monitor Bank 4 Sensor 2 | 0.00 | 1.275 | V | 0.005 Rich to lean sensor threshold voltage |
| 010F |  | O2 Sensor Monitor Bank 4 Sensor 3 | 0.00 | 1.275 | V | 0.005 Rich to lean sensor threshold voltage |
| 0110 |  | O2 Sensor Monitor Bank 4 Sensor 4 | 0.00 | 1.275 | V | 0.005 Rich to lean sensor threshold voltage |
| 0201 |  | O2 Sensor Monitor Bank 1 Sensor 1 | 0.00 | 1.275 | V | 0.005 Lean to Rich sensor threshold voltage |
| 0202 |  | O2 Sensor Monitor Bank 1 Sensor 2 | 0.00 | 1.275 | V | 0.005 Lean to Rich sensor threshold voltage |
| 0203 |  | O2 Sensor Monitor Bank 1 Sensor 3 | 0.00 | 1.275 | V | 0.005 Lean to Rich sensor threshold voltage |
| 0204 |  | O2 Sensor Monitor Bank 1 Sensor 4 | 0.00 | 1.275 | V | 0.005 Lean to Rich sensor threshold voltage |
| 0205 |  | O2 Sensor Monitor Bank 2 Sensor 1 | 0.00 | 1.275 | V | 0.005 Lean to Rich sensor threshold voltage |
| 0206 |  | O2 Sensor Monitor Bank 2 Sensor 2 | 0.00 | 1.275 | V | 0.005 Lean to Rich sensor threshold voltage |
| 0207 |  | O2 Sensor Monitor Bank 2 Sensor 3 | 0.00 | 1.275 | V | 0.005 Lean to Rich sensor threshold voltage |
| 0208 |  | O2 Sensor Monitor Bank 2 Sensor 4 | 0.00 | 1.275 | V | 0.005 Lean to Rich sensor threshold voltage |
| 0209 |  | O2 Sensor Monitor Bank 3 Sensor 1 | 0.00 | 1.275 | V | 0.005 Lean to Rich sensor threshold voltage |
| 020A |  | O2 Sensor Monitor Bank 3 Sensor 2 | 0.00 | 1.275 | V | 0.005 Lean to Rich sensor threshold voltage |
| 020B |  | O2 Sensor Monitor Bank 3 Sensor 3 | 0.00 | 1.275 | V | 0.005 Lean to Rich sensor threshold voltage |
| 020C |  | O2 Sensor Monitor Bank 3 Sensor 4 | 0.00 | 1.275 | V | 0.005 Lean to Rich sensor threshold voltage |
| 020D |  | O2 Sensor Monitor Bank 4 Sensor 1 | 0.00 | 1.275 | V | 0.005 Lean to Rich sensor threshold voltage |
| 020E |  | O2 Sensor Monitor Bank 4 Sensor 2 | 0.00 | 1.275 | V | 0.005 Lean to Rich sensor threshold voltage |
| 020F |  | O2 Sensor Monitor Bank 4 Sensor 3 | 0.00 | 1.275 | V | 0.005 Lean to Rich sensor threshold voltage |
| 0210 |  | O2 Sensor Monitor Bank 4 Sensor 4 | 0.00 | 1.275 | V | 0.005 Lean to Rich sensor threshold voltage |
| **PID (hex)** | **Data bytes returned** | **Description** | **Min value** | **Max value** | **Units** | **Formula**[[a]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-formula-3) |

**Service 09 - Request vehicle information**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=8)]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PID (hex)** | **Data bytes returned** | **Description** | **Min value** | **Max value** | **Units** | **Formula**[[a]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-formula-3) |
| 00 | 4 | Service 9 supported PIDs ($01 to $20) |  |  |  | Bit encoded. [A7..D0] = [PID $01..PID $20] [See below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_01_PID_00) |
| 01 | 1 | VIN Message Count in PID 02. Only for ISO 9141-2, ISO 14230-4 and SAE J1850. |  |  |  | Usually the value will be 5. |
| 02 | 17 | [Vehicle Identification Number](https://en.wikipedia.org/wiki/Vehicle_Identification_Number) (VIN) |  |  |  | 17-char VIN, ASCII-encoded and left-padded with null chars (0x00) if needed to. |
| 03 | 1 | Calibration ID message count for PID 04. Only for ISO 9141-2, ISO 14230-4 and SAE J1850. |  |  |  | It will be a multiple of 4 (4 messages are needed for each ID). |
| 04 | 16,32,48,64.. | Calibration ID |  |  |  | Up to 16 ASCII chars. Data bytes not used will be reported as null bytes (0x00). Several CALID can be outputed (16 bytes each) |
| 05 | 1 | Calibration verification numbers (CVN) message count for PID 06. Only for ISO 9141-2, ISO 14230-4 and SAE J1850. |  |  |  |  |
| 06 | 4,8,12,16 | Calibration Verification Numbers (CVN) Several CVN can be output (4 bytes each) the number of CVN and CALID must match |  |  |  | Raw data left-padded with null characters (0x00). Usually displayed as hex string. |
| 07 | 1 | In-use performance tracking message count for PID 08 and 0B. Only for ISO 9141-2, ISO 14230-4 and SAE J1850. | 8 | 10 |  | 8 if sixteen values are required to be reported, 9 if eighteen values are required to be reported, and 10 if twenty values are required to be reported (one message reports two values, each one consisting in two bytes). |
| 08 | 4 | In-use performance tracking for spark ignition vehicles |  |  |  | 4 or 5 messages, each one containing 4 bytes (two values). [See below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_09_PID_08) |
| 09 | 1 | ECU name message count for PID 0A |  |  |  |  |
| 0A | 20 | ECU name |  |  |  | ASCII-coded. Right-padded with null chars (0x00). |
| 0B | 4 | In-use performance tracking for compression ignition vehicles |  |  |  | 5 messages, each one containing 4 bytes (two values). [See below](https://en.wikipedia.org/wiki/OBD-II_PIDs#Service_09_PID_0B) |
| **PID (hex)** | **Data bytes returned** | **Description** | **Min value** | **Max value** | **Units** | **Formula**[[a]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-formula-3) |

* 1. ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_ref-formula_3-0) [***b***](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_ref-formula_3-1) [***c***](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_ref-formula_3-2) [***d***](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_ref-formula_3-3) [***e***](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_ref-formula_3-4) [***f***](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_ref-formula_3-5) [***g***](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_ref-formula_3-6) [***h***](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_ref-formula_3-7) In the formula column, letters A, B, C, etc. represent the first, second, third, etc. byte of the data. For example, for two data bytes 0F 19, A = 0F and B = 19. Where a (?) appears, contradictory or incomplete information was available.
  2. [**^**](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_ref-6) Starting with MY 2010 the [California Air Resources Board](https://en.wikipedia.org/wiki/California_Air_Resources_Board) mandated that all diesel vehicles must supply total engine hours [[4]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-finalregorder2-5)
  3. [**^**](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_ref-7) Starting with MY 2019 the [California Air Resources Board](https://en.wikipedia.org/wiki/California_Air_Resources_Board) mandated that all vehicles must supply odometer[[4]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-finalregorder2-5)

**Bitwise encoded PIDs**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=9)]

Some of the PIDs in the above table cannot be explained with a simple formula. A more elaborate explanation of these data is provided here:

**Service 01 PID 00 - Show PIDs supported**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=10)]

A request for this PID returns 4 bytes of data ([Big-endian](https://en.wikipedia.org/wiki/Big-endian)). Each bit, from [MSB](https://en.wikipedia.org/wiki/Most_significant_bit) to [LSB](https://en.wikipedia.org/wiki/Least_significant_bit), represents one of the next 32 PIDs and specifies whether that PID is supported.

For example, if the car response is BE1FA813, it can be decoded like this:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Hexadecimal** | B | | | | E | | | | 1 | | | | F | | | | A | | | | 8 | | | | 1 | | | | 3 | | | |
| **Binary** | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| **Supported?** | Yes | No | Yes | Yes | Yes | Yes | Yes | No | No | No | No | Yes | Yes | Yes | Yes | Yes | Yes | No | Yes | No | Yes | No | No | No | No | No | No | Yes | No | No | Yes | Yes |
| **PID number** | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D | 1E | 1F | 20 |

So, supported PIDs are: 01, 03, 04, 05, 06, 07, 0C, 0D, 0E, 0F, 10, 11, 13, 15, 1C, 1F and 20

**Service 01 PID 01 - Monitor status since DTCs cleared**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=11)]

A request for this PID returns 4 bytes of data, labeled A, B, C and D.

The first byte (A) contains two pieces of information. Bit A7 ([MSB](https://en.wikipedia.org/wiki/Most_significant_bit) of byte A) indicates whether or not the MIL (malfunction indicator light, aka. check engine light) is illuminated. Bits A6 through A0 represent the number of diagnostic trouble codes currently flagged in the ECU.

The second, third, and fourth bytes (B, C and D) give information about the availability and completeness of certain on-board tests ("OBD readiness checks"). The third and fourth bytes are to be interpreted differently depending upon whether the engine is [spark](https://en.wikipedia.org/wiki/Spark_plug) [ignition](https://en.wikipedia.org/wiki/Ignition_system) (e.g. Otto or Wankel engines) or [compression ignition](https://en.wikipedia.org/wiki/Diesel_engine) (e.g. Diesel engines). In the second byte (B), bit 3 indicates the engine type and thus how to interpret bytes C and D, with 0 being spark (Otto or Wankel) and 1 (set) being compression (Diesel). Bits B6 to B4 and B2 to B0 are used for information about tests that not engine-type specific, and thus termed *common* tests. Note that for bits indicating test **availability** a bit set to 1 indicates available, whilst for bits indicating test **completeness** a bit set to 0 indicates complete.

|  |  |
| --- | --- |
| **Bits** | **Definition** |
| A7 | State of the CEL/MIL (on/off). |
| A6-A0 | Number of confirmed emissions-related DTCs available for display. |
| B7 | Reserved (should be 0) |
| B6-B4 | Bitmap indicating completeness of *common* tests. |
| B3 | Indication of engine type 0 = Spark ignition (e.g. Otto or Wankel engines) 1 = Compression ignition (e.g. Diesel engines) |
| B2-B0 | Bitmap indicating availability of *common* tests. |
| C7-C0 | Bitmap indicating availability of engine-type specific tests. |
| D7-D0 | Bitmap indicating completeness of engine-type specific tests. |

Bits from byte B representing *common* test indicators (those not engine-type specific) are mapped as follows:

|  |  |  |
| --- | --- | --- |
|  | **Test availability** | **Test completeness** |
| **Components** | B2 | B6 |
| **Fuel System** | B1 | B5 |
| **Misfire** | B0 | B4 |

Bytes C and D are mapped as follows for spark ignition engine types (e.g. Otto or Wankel engines):

|  |  |  |
| --- | --- | --- |
|  | **Test availability** | **Test completeness** |
| **EGR and/or VVT System** | C7 | D7 |
| **Oxygen Sensor Heater** | C6 | D6 |
| **Oxygen Sensor** | C5 | D5 |
| **Gasoline Particulate Filter**[[a]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-10) | C4 | D4 |
| **Secondary Air System** | C3 | D3 |
| **Evaporative System** | C2 | D2 |
| **Heated Catalyst** | C1 | D1 |
| **Catalyst** | C0 | D0 |

Bytes C and D are alternatively mapped as follows for compression ignition engine types (Diesel engines):

|  |  |  |
| --- | --- | --- |
|  | **Test availability** | **Test completeness** |
| **EGR and/or VVT System** | C7 | D7 |
| **PM filter monitoring** | C6 | D6 |
| **Exhaust Gas Sensor** | C5 | D5 |
| **- Reserved -** | C4 | D4 |
| **Boost Pressure** | C3 | D3 |
| **- Reserved -** | C2 | D2 |
| **NOx/SCR Monitor** | C1 | D1 |
| **NMHC Catalyst**[[b]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-11) | C0 | D0 |

* 1. [**^**](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_ref-10) A common misconception is that C4/D4 was A/C Refrigerant, however it had been listed as Reserved in J1979 for years, and was recently defined as GPF.
  2. [**^**](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_ref-11) NMHC *may* stand for Non-Methane HydroCarbons, but J1979 does not enlighten us. The translation would be the ammonia sensor in the SCR catalyst.

**Service 01 PID 41 - Monitor status this drive cycle**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=12)]

A request for this PID returns 4 bytes of data. The data returned is of an identical form to that returned for PID 01, with one exception - the first byte is always zero.

**Service 01 PID 78 and 79 - Exhaust Gas temperature (EGT) Bank 1 and Bank 2**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=13)]

A request for one of these two PIDs will return 9 bytes of data. PID 78 returns data relating to [EGT](https://en.wikipedia.org/wiki/Exhaust_gas_temperature) sensors for bank 1, whilst PID 79 similarly returns data for bank 2. The first byte is a bit encoded field indicating which EGT sensors are supported for the respective bank.

|  |  |
| --- | --- |
| **Bytes** | **Description** |
| A | EGT sensor support |
| B-C | Temperature read by EGT sensor 1 |
| D-E | Temperature read by EGT sensor 2 |
| F-G | Temperature read by EGT sensor 3 |
| H-I | Temperature read by EGT sensor 4 |

The first byte is bit-encoded as follows:

|  |  |
| --- | --- |
| **Bits** | **Description** |
| A7-A4 | Reserved |
| A3 | EGT sensor 4 supported? |
| A2 | EGT sensor 3 supported? |
| A1 | EGT sensor 2 supported? |
| A0 | EGT sensor 1 supported? |

Bytes B through I provide 16-bit integers indicating the temperatures of the sensors. The temperature values are interpreted in degrees Celsius in the range -40 to 6513.5 (scale 0.1), using the usual (A×256+B)/10−40 formula (MSB is A, LSB is B). Only values for which the corresponding sensor is supported are meaningful.

**Service 03 (no PID required) - Show stored Diagnostic Trouble Codes**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=14)]

A request for this service returns a list of the DTCs that have been set. The list is encapsulated using the [ISO 15765-2](https://en.wikipedia.org/wiki/ISO_15765-2) protocol.

If there are two or fewer DTCs (up to 4 bytes) then they are returned in an [ISO-TP](https://en.wikipedia.org/wiki/ISO-TP) Single Frame (SF). Three or more DTCs in the list are reported in multiple frames, with the exact count of frames dependent on the communication type and addressing details.

Each trouble code requires 2 bytes to describe. Encoded in these bytes are a category and a number. It is typically shown decoded into a five-character form like "U0158", where the first character (here 'U') represents the category the DTC belongs to, and the remaining four characters are a hexadecimal representation of the number under that category. The first two bits (A7 and A6) of the first byte (A) represent the category. The remaining 14 bits represent the number. Of note is that since the second character is formed from only two bits, it can thus only be within the range 0-3.

|  |  |
| --- | --- |
| **Bits** | **Definition** |
| A7-A6 | Category 00: **P** - Powertrain 01: **C** - Chassis 10: **B** - Body 11: **U** - Network[[a]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-12) |
| A5-B0 | Number (within category) |

* 1. [**^**](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_ref-12) Whilst this is commonly referred to as the network category, it may originally have been the 'undefined' category, hence the use of the letter 'U' rather than 'N'.

An example DTC of "U0158" would be decoded as follows:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| **Binary** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| **Hexadecimal** | C | | | | 1 | | | | 5 | | | | 8 | | | |
| **Decoded DTC** | U | | 0 | | 1 | | | | 5 | | | | 8 | | | |

The resulting five-character code, e.g. "U0158", can be looked up in a table of OBD-II DTCs to get an actual description of what it represents. Of note, whilst some blocks of DTC code ranges have generic meanings that apply to all vehicles and manufacturers, the meanings of others can vary per manufacturer or even model.

It is also worth noting that DTCs may sometimes be encountered in a four-character form, e.g. "C158", which is simply the plain hexadecimal representation of the two bytes, with proper decoding with respect to the category not having been performed.

**Service 09 PID 08 - In-use performance tracking for spark ignition engines**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=15)]

It provides information about track in-use performance for catalyst banks, oxygen sensor banks, evaporative leak detection systems, [EGR systems](https://en.wikipedia.org/wiki/Exhaust_gas_recirculation) and secondary air system.

The numerator for each component or system tracks the number of times that all conditions necessary for a specific monitor to detect a malfunction have been encountered. The denominator for each component or system tracks the number of times that the vehicle has been operated in the specified conditions.

The count of data items should be reported at the beginning (the first byte).

All data items of the In-use Performance Tracking record consist of two bytes and are reported in this order (each message contains two items, hence the message length is 4).

|  |  |
| --- | --- |
| **Mnemonic** | **Description** |
| OBDCOND | OBD Monitoring Conditions Encountered Counts |
| IGNCNTR | Ignition Counter |
| CATCOMP1 | Catalyst Monitor Completion Counts Bank 1 |
| CATCOND1 | Catalyst Monitor Conditions Encountered Counts Bank 1 |
| CATCOMP2 | Catalyst Monitor Completion Counts Bank 2 |
| CATCOND2 | Catalyst Monitor Conditions Encountered Counts Bank 2 |
| O2SCOMP1 | O2 Sensor Monitor Completion Counts Bank 1 |
| O2SCOND1 | O2 Sensor Monitor Conditions Encountered Counts Bank 1 |
| O2SCOMP2 | O2 Sensor Monitor Completion Counts Bank 2 |
| O2SCOND2 | O2 Sensor Monitor Conditions Encountered Counts Bank 2 |
| EGRCOMP | EGR Monitor Completion Condition Counts |
| EGRCOND | EGR Monitor Conditions Encountered Counts |
| AIRCOMP | AIR Monitor Completion Condition Counts (Secondary Air) |
| AIRCOND | AIR Monitor Conditions Encountered Counts (Secondary Air) |
| EVAPCOMP | EVAP Monitor Completion Condition Counts |
| EVAPCOND | EVAP Monitor Conditions Encountered Counts |
| SO2SCOMP1 | Secondary O2 Sensor Monitor Completion Counts Bank 1 |
| SO2SCOND1 | Secondary O2 Sensor Monitor Conditions Encountered Counts Bank 1 |
| SO2SCOMP2 | Secondary O2 Sensor Monitor Completion Counts Bank 2 |
| SO2SCOND2 | Secondary O2 Sensor Monitor Conditions Encountered Counts Bank 2 |

**Service 09 PID 0B - In-use performance tracking for compression ignition engines**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=16)]

It provides information about track in-use performance for NMHC catalyst, NOx catalyst monitor, NOx adsorber monitor, PM filter monitor, exhaust gas sensor monitor, EGR/ VVT monitor, boost pressure monitor and fuel system monitor.

All data items consist of two bytes and are reported in this order (each message contains two items, hence message length is 4):

|  |  |
| --- | --- |
| **Mnemonic** | **Description** |
| OBDCOND | OBD Monitoring Conditions Encountered Counts |
| IGNCNTR | Ignition Counter |
| HCCATCOMP | NMHC Catalyst Monitor Completion Condition Counts |
| HCCATCOND | NMHC Catalyst Monitor Conditions Encountered Counts |
| NCATCOMP | NOx/SCR Catalyst Monitor Completion Condition Counts |
| NCATCOND | NOx/SCR Catalyst Monitor Conditions Encountered Counts |
| NADSCOMP | NOx Adsorber Monitor Completion Condition Counts |
| NADSCOND | NOx Adsorber Monitor Conditions Encountered Counts |
| PMCOMP | PM Filter Monitor Completion Condition Counts |
| PMCOND | PM Filter Monitor Conditions Encountered Counts |
| EGSCOMP | Exhaust Gas Sensor Monitor Completion Condition Counts |
| EGSCOND | Exhaust Gas Sensor Monitor Conditions Encountered Counts |
| EGRCOMP | EGR and/or VVT Monitor Completion Condition Counts |
| EGRCOND | EGR and/or VVT Monitor Conditions Encountered Counts |
| BPCOMP | Boost Pressure Monitor Completion Condition Counts |
| BPCOND | Boost Pressure Monitor Conditions Encountered Counts |
| FUELCOMP | Fuel Monitor Completion Condition Counts |
| FUELCOND | Fuel Monitor Conditions Encountered Counts |

**Enumerated PIDs**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=17)]

Some PIDs are to be interpreted specially, and aren't necessarily exactly bitwise encoded, or in any scale. The values for these PIDs are [enumerated](https://en.wikipedia.org/wiki/Enumerated_type).

**Service 01 PID 03 - Fuel system status**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=18)]

A request for this PID returns 2 bytes of data. The first byte describes fuel system #1. The second byte describes fuel system #2 (if it exists) and is encoded identically to the first byte. The meaning assigned to the value of each byte is as follows:

|  |  |
| --- | --- |
| **Value** | **Description** |
| 0 | The motor is off |
| 1 | Open loop due to insufficient engine temperature |
| 2 | Closed loop, using oxygen sensor feedback to determine fuel mix |
| 4 | Open loop due to engine load OR fuel cut due to deceleration |
| 8 | Open loop due to system failure |
| 16 | Closed loop, using at least one oxygen sensor but there is a fault in the feedback system |

Any other value is an invalid response.

**Service 01 PID 12 - Commanded secondary air status**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=19)]

A request for this PID returns a single byte of data which describes the secondary air status.

|  |  |
| --- | --- |
| **Value** | **Description** |
| 1 | Upstream |
| 2 | Downstream of catalytic converter |
| 4 | From the outside atmosphere or off |
| 8 | Pump commanded on for diagnostics |

Any other value is an invalid response.

**Service 01 PID 1C - OBD standards this vehicle conforms to**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=20)]

A request for this PID returns a single byte of data which describes which OBD standards this ECU was designed to comply with. The different values the data byte can hold are shown below, next to what they mean:

|  |  |
| --- | --- |
| **Value** | **Description** |
| 1 | OBD-II as defined by the [CARB](https://en.wikipedia.org/wiki/California_Air_Resources_Board) |
| 2 | OBD as defined by the [EPA](https://en.wikipedia.org/wiki/United_States_Environmental_Protection_Agency) |
| 3 | OBD and OBD-II |
| 4 | OBD-I |
| 5 | Not OBD compliant |
| 6 | EOBD (Europe) |
| 7 | EOBD and OBD-II |
| 8 | EOBD and OBD |
| 9 | EOBD, OBD and OBD II |
| 10 | JOBD (Japan) |
| 11 | JOBD and OBD II |
| 12 | JOBD and EOBD |
| 13 | JOBD, EOBD, and OBD II |
| 14 | Reserved |
| 15 | Reserved |
| 16 | Reserved |
| 17 | Engine Manufacturer Diagnostics (EMD) |
| 18 | Engine Manufacturer Diagnostics Enhanced (EMD+) |
| 19 | Heavy Duty On-Board Diagnostics (Child/Partial) (HD OBD-C) |
| 20 | Heavy Duty On-Board Diagnostics (HD OBD) |
| 21 | World Wide Harmonized OBD (WWH OBD) |
| 22 | Reserved |
| 23 | Heavy Duty Euro OBD Stage I without NOx control (HD EOBD-I) |
| 24 | Heavy Duty Euro OBD Stage I with NOx control (HD EOBD-I N) |
| 25 | Heavy Duty Euro OBD Stage II without NOx control (HD EOBD-II) |
| 26 | Heavy Duty Euro OBD Stage II with NOx control (HD EOBD-II N) |
| 27 | Reserved |
| 28 | Brazil OBD Phase 1 (OBDBr-1) |
| 29 | Brazil OBD Phase 2 (OBDBr-2) |
| 30 | Korean OBD (KOBD) |
| 31 | India OBD I (IOBD I) |
| 32 | India OBD II (IOBD II) |
| 33 | Heavy Duty Euro OBD Stage VI (HD EOBD-IV) |
| 34-250 | Reserved |
| 251-255 | Not available for assignment (SAE [J1939](https://en.wikipedia.org/wiki/J1939) special meaning) |

**Service 01 PID 51 - Fuel Type Coding**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=21)]

This PID returns a value from an enumerated list giving the fuel type of the vehicle. The fuel type is returned as a single byte, and the value is given by the following table:

|  |  |
| --- | --- |
| **Value** | **Description** |
| 0 | Not available |
| 1 | Gasoline |
| 2 | Methanol |
| 3 | Ethanol |
| 4 | Diesel |
| 5 | [LPG](https://en.wikipedia.org/wiki/Liquefied_petroleum_gas) |
| 6 | [CNG](https://en.wikipedia.org/wiki/Compressed_natural_gas) |
| 7 | Propane |
| 8 | Electric |
| 9 | [Bifuel](https://en.wikipedia.org/wiki/Bi-fuel_vehicle) running Gasoline |
| 10 | Bifuel running Methanol |
| 11 | Bifuel running Ethanol |
| 12 | Bifuel running LPG |
| 13 | Bifuel running CNG |
| 14 | Bifuel running Propane |
| 15 | Bifuel running Electricity |
| 16 | Bifuel running electric and combustion engine |
| 17 | Hybrid gasoline |
| 18 | Hybrid Ethanol |
| 19 | Hybrid Diesel |
| 20 | Hybrid Electric |
| 21 | Hybrid running electric and combustion engine |
| 22 | Hybrid Regenerative |
| 23 | Bifuel running diesel |

Any other value is reserved by ISO/SAE. There are currently no definitions for [flexible-fuel vehicle](https://en.wikipedia.org/wiki/Flexible-fuel_vehicle).

**Non-standard PIDs**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=22)]

The majority of all OBD-II PIDs in use are non-standard. For most modern vehicles, there are many more functions supported on the OBD-II interface than are covered by the standard PIDs, and there is relatively minor overlap between vehicle manufacturers for these non-standard PIDs.

There is very limited information available in the public domain for non-standard PIDs. The primary source of information on non-standard PIDs across different manufacturers is maintained by the US-based [Equipment and Tool Institute](https://en.wikipedia.org/wiki/Equipment_and_Tool_Institute) and only available to members. The price of ETI membership for access to scan codes varies based on company size defined by annual sales of automotive tools and equipment in North America:

|  |  |
| --- | --- |
| **Annual Sales in North America** | **Annual Dues** |
| Under $10,000,000 | $5,000 |
| $10,000,000 - $50,000,000 | $7,500 |
| Greater than $50,000,000 | $10,000 |

However, even ETI membership will not provide full documentation for non-standard PIDs. ETI states:[[7]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-eti-faq-13)[[8]](https://en.wikipedia.org/wiki/OBD-II_PIDs#cite_note-14)

Some OEMs refuse to use ETI as a one-stop source of scan tool information. They prefer to do business with each tool company separately. These companies also require that you enter into a contract with them. The charges vary but here is a snapshot as of April 13th, 2015 of the per year charges:

|  |  |
| --- | --- |
| GM | $50,000 |
| Honda | $5,000 |
| Suzuki | $1,000 |
| BMW | $25,500 plus $2,000 per update. Updates occur annually. |

**CAN (11-bit) bus format**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=23)]

As defined in ISO 15765-4, emissions protocols (including OBD-II, EOBD, UDS, etc.) use the ISO-TP transport layer (ISO 15765-2). All CAN frames sent using ISO-TP use a data length of 8 bytes (and DLC of 8). It is recommended to pad the unused data bytes with 0xCC.

The PID query and response occurs on the vehicle's CAN bus. Standard OBD requests and responses use functional addresses. The diagnostic reader initiates a query using CAN ID 7DFh, which acts as a broadcast address, and accepts responses from any ID in the range 7E8h to 7EFh. ECUs that can respond to OBD queries listen both to the functional broadcast ID of 7DFh and one assigned ID in the range 7E0h to 7E7h. Their response has an ID of their assigned ID plus 8 e.g. 7E8h through 7EFh.

This approach allows up to eight ECUs, each independently responding to OBD queries. The diagnostic reader can use the ID in the ECU response frame to continue communication with a specific ECU. In particular, multi-frame communication requires a response to the specific ECU ID rather than to ID 7DFh.

CAN bus may also be used for communication beyond the standard OBD messages. Physical addressing uses particular CAN IDs for specific modules (e.g., 720h for the instrument cluster in Fords) with proprietary frame payloads.

**Query**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=24)]

The functional PID query is sent to the vehicle on the CAN bus at ID 7DFh, using 8 data bytes. The bytes are:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Byte** | | | | | | | |
| **PID Type** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| SAE Standard | Number of additional data bytes: 2 | Service 01 = show current data; 02 = freeze frame; etc. | PID code (e.g.: 05 = Engine coolant temperature) | not used ([ISO 15765-2](https://en.wikipedia.org/wiki/ISO_15765-2) suggests CCh) | | | | |
| Vehicle specific | Number of additional data bytes: 3 | Custom service: (e.g.: 22 = enhanced data) | PID code (e.g.: 4980h) | | not used ([ISO 15765-2](https://en.wikipedia.org/wiki/ISO_15765-2) suggests CCh) | | | |

**Response**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=25)]

The vehicle responds to the PID query on the CAN bus with message IDs that depend on which module responded. Typically the engine or main ECU responds at ID 7E8h. Other modules, like the hybrid controller or battery controller in a Prius, respond at 07E9h, 07EAh, 07EBh, etc. These are 8h higher than the physical address the module responds to. Even though the number of bytes in the returned value is variable, the message uses 8 data bytes regardless ([CAN bus](https://en.wikipedia.org/wiki/CAN_bus) protocol form Frameformat with 8 data bytes). The bytes are:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Byte** | | | | | | | |
| **CAN Address** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| SAE Standard 7E8h, 7E9h, 7EAh, etc. | Number of additional data bytes: 3 to 6 | Custom service Same as query, except that 40h is added to the service value. So: 41h = show current data; 42h = freeze frame; etc. | PID code (e.g.: 05 = Engine coolant temperature) | value of the specified parameter, byte 0 | value, byte 1 (optional) | value, byte 2 (optional) | value, byte 3 (optional) | not used (may be 00h or 55h) |
| Vehicle specific 7E8h, or 8h + physical ID of module. | Number of additional data bytes: 4to 7 | Custom service: same as query, except that 40h is added to the service value.(e.g.: 62h = response to service 22h request) | PID code (e.g.: 4980h) | | value of the specified parameter, byte 0 | value, byte 1 (optional) | value, byte 2 (optional) | value, byte 3 (optional) |
| Vehicle specific 7E8h, or 8h + physical ID of module. | Number of additional data bytes: 3 | 7Fh this a general response usually indicating the module doesn't recognize the request. | Custom service: (e.g.: 22h = enhanced diagnostic data by PID, 21h = enhanced data by offset) | 31h | not used (may be 00h) | | | |

**See also**

[[edit](https://en.wikipedia.org/w/index.php?title=OBD-II_PIDs&action=edit&section=26)]

* [Engine control unit](https://en.wikipedia.org/wiki/Engine_control_unit)
* [ELM327](https://en.wikipedia.org/wiki/ELM327), a very common microcontroller (silicon chip) and multi-protocol interpreter used in OBD-II vehicle communication interfaces
* [Unified Diagnostic Services](https://en.wikipedia.org/wiki/Unified_Diagnostic_Services), the ISO standard

---------------------------------------------------------------------------------------------------------------------

**Шина CAN – Введение**

Протокол CAN является стандартом ISO (ISO 11898) в области последовательной передачи данных. Протокол был разработан с прицелом на использование в транспортных приложениях. Сегодня CAN получил широкое распространение и используется в системах автоматизации промышленного производства, а также на транспорте.

Стандарт CAN состоит из физического уровня и уровня передачи данных, определяющего несколько различных типов сообщений, правила разрешения конфликтов при доступе к шине и защиту от сбоев.

**Протокол CAN**

Протокол CAN описан в стандарте ISO 11898–1 и может быть кратко охарактеризован следующим образом:

• физический уровень использует дифференциальную передачу данных по витой паре;

• для управления доступом к шине используется неразрушающее bit–wise разрешение конфликтов;

• сообщения имеют малые размеры (по большей части 8 байт данных) и защищены контрольной суммой;

• в сообщениях отсутствуют явные адреса, вместо этого каждое сообщение содержит числовое значение, которое управляет его очередностью на шине, а также может служить идентификатором содержимого сообщения;

• продуманная схема обработки ошибок, обеспечивающая повторную передачу сообщений, если они не были получены должным образом;  
• имеются эффективные средства для изоляции сбоев и удаления сбойных узлов с шины.

**Протоколы более высоких уровней**

Сам по себе протокол CAN определяет всего лишь, как малые пакеты данных можно безопасно переместить из точки A в точку B посредством коммуникационной среды. Он, как и следовало ожидать, ничего не говорит о том, как управлять потоком; передавать большое количество данных, нежели помещается в 8–байтное сообщение; ни об адресах узлов; установлении соединения и т.п. Эти пункты определяются протоколом более высокого уровня (Higher Layer Protocol, HLP). Термин HLP происходит из модели OSI и её семи уровней.

Протоколы более высокого уровня используются для:

• стандартизации процедуры запуска, включая выбор скорости передачи данных;

• распределения адресов среди взаимодействующих узлов или типов сообщений;

• определения разметки сообщений;  
• обеспечения порядка обработки ошибок на уровне системы.

**Пользовательские группы и т.п.**

Одним из наиболее эффективных способов повышения вашей компетентности в области CAN является участие в работе, осуществляемой в рамках существующих пользовательских групп. Даже если вы не планируете активно участвовать в работе, пользовательские группы могут являться хорошим источником информации. Посещение конференций является ещё одним хорошим способом получения исчерпывающей и точной информации.

[• Пользовательские группы и прочее из области CAN >>](http://www.kvaser.com/can/info/main.htm#usergroup)

**Продукты CAN**

На низком уровне принципиально различают два типа продуктов CAN, доступных на открытом рынке – микросхемы CAN и инструменты разработки CAN. На более высоком уровне – другие два типа продуктов: модули CAN и инструменты проектирования CAN. Широкий спектр данных продуктов доступен на открытом рынке в настоящее время.

[• Ознакомьтесь с нашим перечнем продуктов CAN >>](https://www.micromax.ru/catalog/?vendor=170&filter=Y)

**Патенты в области CAN**

Патенты, относящиеся к приложениям CAN, могут быть различных типов: реализация синхронизации и частот, передача больших наборов данных (в протоколе CAN используются кадры данных длиной всего лишь 8 байт) и т.п.

**Системы распределённого управления**

Протокол CAN является хорошей основой для разработки систем распределённого управления. Метод разрешения конфликтов, используемый CAN, обеспечивает то, что каждый узел CAN будет взаимодействовать с теми сообщениями, которые относятся к данному узлу.

Систему распределённого управления можно описать как систему, вычислительная мощность которой распределена между всеми узлами системы. Противоположный вариант – система с центральным процессором и локальными точками ввода–вывода.

**Сообщения CAN**

Шина CAN относится к широковещательным шинам. Это означает, что все узлы могут «слушать» все передачи. Не существует возможности послать сообщение конкретному узлу, все без исключения узлы будут принимать все сообщения. Оборудование CAN, однако, обеспечивает возможность локальной фильтрации, так что каждый модуль может реагировать только на интересующее его сообщение.

**Адресация сообщений CAN**

CAN использует относительно короткие сообщения – максимальная длина информационного поля составляет 94 бита. В сообщениях отсутствует явный адрес, их можно назвать контентно–адрессованными: содержимое сообщения имплицитно (неявным образом) определяет адресата.

**Типы сообщений**

Существует 4 типа сообщений (или кадров), передающихся по шине CAN:

• кадр данных (Data Frame);

• удаленный кадр (Remote Frame);

• кадр ошибки (Error Frame);

• кадр перегрузки (Overload Frame).

**Кадр данных**

Кратко: «Всем привет, есть данные с маркировкой X, надеюсь вам понравятся!»  
Кадр данных – самый распространенный тип сообщения. Он содержит в себе следующие основные части (некоторые детали не рассматриваются для краткости):

• Поле арбитража (Arbitration Field), которое определяет очередность сообщения в том случае, когда за шину борятся два или более узла. Поле арбитража содержит:

• В случае CAN 2.0A, 11–битный идентификатор и один бит, бит RTR который является определяющим для кадров данных.

• В случае CAN 2.0B, 29–битный идентификатор (который также содержит два рецессивных бита: SRR и IDE) и бит RTR.

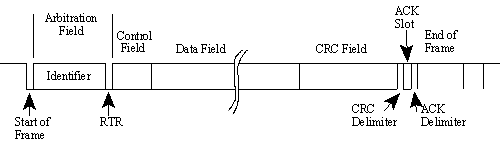
• Поле данных (Data Field), которое содержит от 0 до 8 байт данных.

• Поле CRC (CRC Field), содержащее 15–битную контрольную сумму, посчитанную для большинства частей сообщения. Эта контрольная сумма используется для обнаружения ошибок.

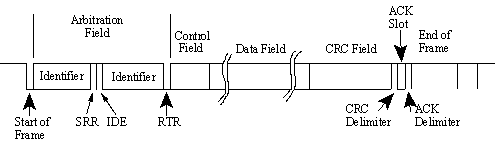
• Слот распознавания (Acknowledgement Slot). Каждый контроллер CAN, способный корректно получить сообщение, посылает бит распознавания (Acknowledgement bit) в конце каждого сообщения. Приемопередатчик проверяет наличие бита распознавания и, если таковой не обнаруживается, высылает сообщение повторно.

Примечание 1: Присутствие на шине бита распознавания не значит ничего, кроме того, что каждый запланированный адресат получил сообщение. Единственное, что становится известно, это факт корректного получения сообщения одним или несколькими узлами шины.

Примечание 2: Идентификатор в поле арбитража, несмотря на свое название, необязательно идентифицирует содержимое сообщения.



*Кадр данных CAN 2.0B («cтандартный CAN»).*



*Кадр данных CAN 2.0B («расширенный CAN»).*

**Удаленный кадр**

Кратко: «Всем привет, кто–нибудь может произвести данные с маркировкой X?»  
Удаленный кадр очень похож на кадр данных, но с двумя важными отличиями:

• он явно помечен как удаленный кадр (бит RTR в поле арбитража является рецессивным), и

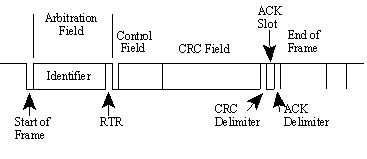
• отсутствует поле данных.

Основной задачей удаленного кадра является запрос на передачу надлежащего кадра данных. Если, скажем, узел A пересылает удаленный кадр с параметром поля арбитража равным 234, то узел B, если он должным образом инициализирован, должен выслать в ответ кадр данных с параметром поля арбитража также равным 234.

Удаленные кадры можно использовать для реализации управления трафиком шины типа «запрос–ответ». На практике, однако, удаленный кадр используется мало. Это не так важно, поскольку стандарт CAN не предписывает действовать именно так, как здесь обозначено. Большинство контроллеров CAN можно запрограммировать так, что они будут автоматически отвечать на удаленный кадр, или же вместо этого извещать локальный процессор.

Есть одна уловка, связанная с удаленным кадром: код длины данных (Data Length Code) должен быть установлен длине ожидаемого ответного сообщения. В противном случае разрешение конфликтов работать не будет.

Иногда требуется чтобы узел, отвечающий на удаленный кадр, начинал свою передачу как только распознавал идентификатор, таким образом «заполняя» пустой удаленный кадр. Это другой случай.

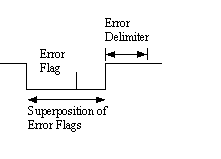


**Кадр ошибки (Error Frame)**

Кратко (все вместе, громко): «О, ДОРОГОЙ, ДАВАЙ ПОПРОБУЕМ ЕЩЁ РАЗОК»  
Кадр ошибки (Error Frame) – это специальное сообщение, нарушающее правила формирования кадров сообщения CAN. Он посылается, когда узел обнаруживает сбой и помогает остальным узлам обнаружить сбой – и они тоже будут отправлять кадры ошибок. Передатчик автоматически попробует послать сообщение повторно. Наличествует продуманная схема счетчиков ошибок, гарантирующая, что узел не сможет нарушить передачу данных по шине путём повторяющейся отсылки кадров ошибки.

Кадр ошибки содержит флаг ошибки (Error Flag), который состоит из 6 бит одинакового значения (таким образом нарушая правило вставки битов) и разграничителя ошибки (Error Delimiter), состоящего из 8 рецессивных бит. Разраничитель ошибки предоставляет некоторое пространство, в котором другие узлы шины могут отправлять свои флаги ошибки после того, как сами обнаружат первый флаг ошибки.

*Вот флаг ошибки:*



**Кадр перегрузки (Overload Frame)**

Кратко: «Я очень занятой 82526 маленький, не могли бы вы подождать минуточку?»  
Кадр перегрузки упоминается здесь лишь для полноты картины. По формату он очень похож на кадр ошибки и передается занятым узлом. Кадр перегрузки используется нечасто, т.к. современные контроллеры CAN достаточно производительны, чтобы его не использовать. Фактически, единственный контроллер, который будет генерировать кадры перегрузки – это ныне устаревший 82526.

**Стандартный и расширенный CAN**

Изначально стандарт CAN установил длину идентификатора в поле арбитража равной 11 битам. Позже, по требованию покупателей стандарт был расширен. Новый формат часто называют расширенным CAN (Extended CAN), он позволяет использовать не менее 29 бит в идентификаторе. Для различения двух типов кадров используется зарезервированный бит в поле управления Control Field.

Формально стандарты именуются следующим образом –

• 2.0A – только с 11–битными идентификаторами;  
• 2.0B – расширенная версия с 29–битными или 11–битными идентификаторами (их можно смешивать). Узел 2.0B может быть

• 2.0B active (активным), т.е. способным передавать и получать расширенные кадры, или

• 2.0B passive (пассивным), т.е. он будет молча сбрасывать полученные расширенные кадры (но, смотрите ниже).

• 1.x – относится к оргинальной спецификации и её ревизиям.

В настоящее время новые контроллеры CAN обычно относятся к типу 2.0B. Контроллер типа 1.x или 2.0A прибудет в замешательство, получив сообщения с 29 битами арбитража. Контроллер 2.0B пассивного типа примет их, опознает, если они верны и, затем – сбросит; a контроллер 2.0B активного типа сможет и передавать, и получать такие сообщения.

Контроллеры 2.0B и 2.0A (равно, как и 1.x) совместимы. Можно использовать их все на одной шине до тех пор, пока контроллеры 2.0B будут воздерживаться от рассылки расширенных кадров.

Иногда люди заявляют, что стандартный CAN «лучше» расширенного CAN, потому что в сообщениях расширенного CAN больше служебных данных. Это необязательно так. Если вы используете поле арбитража для передачи данных, то кадр расширенного CAN может содержать меньше служебных данных, чем кадр стандартного CAN.

**Основной CAN (Basic CAN) и полный CAN (Full CAN)**

Термины Basic CAN и Full CAN берут начало в «детстве» CAN. Когда–то существовал CAN–контроллер Intel 82526, предоставлявший программисту интерфейс в стиле DPRAM. Потом появился Philips с моделью 82C200, в котором применялась FIFO–ориентированная модель программирования и ограниченные возможности фильтрации. Для обозначения различия между двумя моделями программирования, люди стали называть способ Intel – Full CAN, а способ Philips – Basic CAN. Сегодня большинство контроллеров CAN поддерживают обе модели программирования, поэтому нет смысла в использовании терминов Full CAN и Basic CAN – фактически, эти термины могут вызвать неразбериху и стоит воздержаться от их употребления.

В действительности, контроллер Full CAN может взаимодействовать с контроллером Basic CAN и наоборот. Проблемы с совместимостью отсутствуют.

**Разрешение конфликтов на шине и приоритет сообщения**

Разрешение конфликтов сообщений (процесс, в результате которого два или более контроллера CAN решают, кто будет пользоваться шиной) очень важно для определения реальной доступности полосы пропускания для передачи данных.

Любой контроллер CAN может начать передачу, когда обнаружит, что шина простаивает. Это может привести к тому, что два или более контроллеров начнут передачу сообщения (почти) одновременно. Конфликт решается следующим образом. Передающие узлы осуществляют мониторинг шины в процессе отправки сообщения. Если узел обнаруживает доминантный уровень в то время, как сам он отправляет рецессивный уровень, он незамедлительно устранится от процесса разрешения конфликта и станет приемником. Разрешение конфликтов осуществляется по всему полю арбитража, и после того, как это поле отсылается, на шине остается только один передатчик. Данный узел продолжит передачу, если ничего не случится. Остальные потенциальные передатчики попытаются передать свои сообщения позже, когда шина освободится. В процессе разрешения конфликта время не теряется.

Важным условием для благополучного разрешения конфликта является невозможность ситуации, при которой два узла могут передать одинаковое поле арбитража. Из этого правила есть одно исключение: если сообщение не содержит данных, то любой узел может передавать это сообщение.

Поскольку, CAN–шина является шиной с подсоединением устройств по типу «монтажное И» (wired–AND) и доминантный бит (Dominant bit) является логическим 0, следовательно сообщение с самым низким в численном выражении полем арбитража выиграет в разрешении конфликта.

Вопрос: Что произойдет в случае, если единственный узел шины попытается отослать сообщение?

Ответ: Узел, разумеется, выиграет в разрешении конфликта и успешно проведет передачу сообщения. Но когда наступит время распознавания… ни один узел не отправит доминантный бит области распознавания, поэтому передатчик определит ошибку распознавания, пошлет флаг ошибки, повысит значение своего счетчика ошибок передачи на 8 и начнет повторную передачу. Этот цикл повторится 16 раз, затем передатчик перейдет в статус пассивной ошибки. В соответствии со специальным правилом в алгоритме ограничения ошибок, значение счетчика ошибок передачи не будет более повышаться, если узел имеет статус пассивной ошибки и ошибка является ошибкой распознавания. Поэтому узел будет осуществлять передачу вечно, до тех пор, пока кто–нибудь не распознает сообщение.

**Адресация и идентификация сообщения**

Повторимся, нет ничего страшного в том, что в сообщениях CAN нет точных адресов. Каждый контроллер CAN будет получать весь траффик шины, и при помощи комбинации аппаратных фильтров и ПО, определять – «интересует» его это сообщение, или нет.

Фактически, в протоколе CAN отсутствует понятие адреса сообщения. Вместо этого содержимое сообщения определяется идентификатором, который находится где–то в сообщении. Сообщения CAN можно назвать «контентно–адрессовнными».

Определённый адрес работает так: «Это сообщение для узла X». Контентно–адресованное сообщение можно описать так: «Это сообщение содержит данные с маркировкой X». Разница между этими двумя концепциями мала, но существенна.

Содержимое поле арбитража используется, в соответствии со стандартом, для определения очередности сообщения на шине. Все контроллеры CAN будут также использовать всё (некоторые – только часть) поле арбитража в качестве ключа в процессе аппаратной фильтрации.

Стандарт не говорит, что поле арбитража непременно должно использоваться в качестве идентификатора сообщения. Тем не менее, это очень распространенный вариант использования.

**Примечание о значениях идентификатора**

Мы говорили, что идентификатору доступны 11 (CAN 2.0A) или 29 (CAN 2.0B) бит. Это не совсем верно. Для совместимости с определенным старым контроллером CAN (угадайте каким?), идентификаторы не должны иметь 7 старших бит установленных в логическую единицу, поэтому 11–битным идентификаторам доступны значения 0..2031, а пользователи 29–битных идентификаторов могут использовать 532676608 различных значений.

Заметьте, что все остальные контроллеры CAN принимают «неправильные» идентификаторы, поэтому в современных системах CAN идентификаторы 2032..2047 могут использоваться без ограничений.

**Физические уровни CAN**

**Шина CAN**

Шина CAN использует код без возвращения к нулю (NRZ) с вставкой битов. Существуют два разных состояния сигнала: доминантное (логический 0) и рецессивное (логическая 1). Они соответствуют определенным электрическим уровням, зависящим от используемого физического уровня (их несколько). Модули подключены к шине по схеме «монтажное И» (wired–AND): если хотя бы один узел переводит шину в доминантное состояние, то вся шина находится в этом состоянии, вне зависмости от того, сколько узлов передают рецессивное состояние.

**Различные физические уровни**

*Физический уровень* определяет электрические уровни и схему передачи сигналов по шине, полное сопротивление кабеля и т.п.

Существует несколько различных версий физических уровней: • Наиболее распространенным является вариант, определенный стандартом CAN, часть ISO 11898–2, и представляющий собой двухпроводную сбалансированную сигнальную схему. Он также иногда называется high–speed CAN.

• Другая часть того же стандарта ISO 11898–3 описывает другую двухпроводную сбалансированную сигнальную схему – для менее скоростной шины. Она устойчива к сбоям, поэтому передача сигналов может продолжаться даже в том случае, когда один из проводов будет перерезан, замкнут на «землю» или в состоянии Vbat. Иногда такая схема называется low–speed CAN.

• SAE J2411 описывает однопроводной (плюс «земля», разумеется) физический уровень. Он используется в основном в автомобилях – например GM–LAN.

• Существуют несколько проприетарных физических уровней.

• В былые времена, когда драйверов CAN не существовало, использовались модификации RS485.

[• Несколько изображений с осциллографа поясняющие сообщения в деталях >>.](http://www.kvaser.com/can/protocol/can_scope_pictures.htm)

Различные физические уровни как правило не могут взаимодействовать между собой. Некоторые комбинации могут работать (или будет казаться, что они работают) в хороших условиях. Например, приемопередатчики high–speed и low–speed могут работать на одной шине лишь иногда.

Абсолютное большинство микросхем приемопередатчиков CAN произведено компанией Philips; в число других производителей входят Bosch, Infineon, Siliconix и Unitrode.

Наиболее распространен приемопередатчик 82C250, в котором реализован физический уровень, описываемый стандартом ISO 11898. Усовершенствованная версия – 82C251.

Распространенный приемопередатчик для «low–speed CAN» – Philips TJA1054.

**Максимальная скорость передачи данных по шине**

Максимальная скорость передачи данных по шине CAN,*в соответствии со стандартом*, равна 1 Мбит/с. Однако некоторые контроллеры CAN поддерживают скорости выше 1 Мбит/с и могут быть использованы в специализированных приложениях.

Low–speed CAN (ISO 11898–3, см. выше) работает на скоростях до 125 кбит/с.

Однопроводная шина CAN в стандартном режиме может передавать данные со скоростью порядка 50 кбит/с, а в специальном высокоскоростном режиме, например для программирования ЭБУ (ECU), около 100 кбит/с.

**Минимальная скорость передачи данных по шине**

Имейте в виду, что некоторые приемопередатчики не позволят вам выбрать скорость ниже определенного значения. Например, при использовании 82C250 или 82C251 вы можете без проблем установить скорость 10 кбит/с, но если вы используете TJA1050, то не сможете установить скорость ниже 50 кбит/с. Сверяйтесь со спецификацией.

**Максимальная длина кабеля**

При скорости передачи данных 1 Мбит/с, максимальная длина используемого кабеля может составлять порядка 40 метров. Это связано с требованием схемы разрешения конфликтов, согласно которому фронт волны сигнала должен иметь возможность дойти до самого дальнего узла и вернуться назад прежде чем бит будет считан. Иными словами, длина кабеля ограничена скоростью света. Предложения по увеличению скорости света рассматривались, но были отвергнуты в связи с межгалактическими проблемами.

Другие максимальные длины кабеля (значения приблизительные):

• 100 метров при 500 кбит/с;

• 200 метров при 250 кбит/с;

• 500 метров при 125 кбит/с;  
• 6 километров при 10 кбит/с.

Если для обеспечения гальванической изоляции используются оптопары, максимальная длина шины соответственно сокращается. Совет: используйте быстрые оптопары, и смотрите на задержку сигнала в устройстве, а не на максимальную скорость передачи данных в спецификации.

**Оконечное прерывание шины**

Шина CAN стандарта ISO 11898 должна заканчиваться терминатором. Это достигается путем установки резистора сопротивлением 120 Ом на каждом конце шины. Терминирование служит двум целям:

1. Убрать отражения сигнала на конце шины.

2. Убедиться, что получает корректные уровни постоянного тока (DC).

Шина CAN стандарта ISO 11898 обязательно должна терминироваться вне зависимости от её скорости. Я повторю: шина CAN стандарта ISO 11898 обязательно должна терминироваться вне зависимости от её скорости. Для лабораторной работы может хватить и одного терминатора. Если ваша шина CAN работает даже при отсутствии терминаторов – вы просто счастливчик.

Заметьте, что *другие физические уровни*, такие как low–speed CAN, однопроводная шина CAN и другие, могут требовать, а могут и не требовать наличия оконечного терминатора шины. Но ваша высокоскоростная шина CAN стандарта ISO 11898 всегда будет требовать наличия хотя бы одного терминатора.

**Кабель**

Стандарт ISO 11898 предписывает, что волновое сопротивление кабеля номинально должно равнятся 120 Ом, однако допускается интервал значений сопротивления [108..132] Ом.

Немногие, из присутствующих сегодня на рынке, кабели удовлетворяют этим требованиям. Есть большая вероятность, что интервал значений сопротивления будет расширен в будущем.

ISO 11898 описывает витую пару, экранированную или неэкранированную. Идёт работа над стандартом однопроводного кабеля SAE J2411.

**Разъемы CAN**

Для разъемов CAN стандартов не существует! Обычно, каждый (!) протокол более высокого уровня (Higher Layer Protocol) описывает один или несколько предпочтительных типов разъемов. Основные типы:

• 9–контактный DSUB, предложен CiA;

• 5–контактный Mini–C и/или Micro–C, используется DeviceNet и SDS;

• 6–контактный Deutsch разъем, предложенный CANHUG для транспортных гидравлических систем.

**Разъемы CAN**

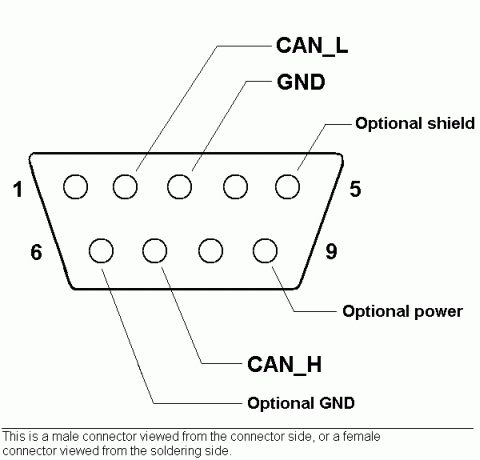
Данное назначение контактов разъема рекомендовано [CiA](http://can-cia.de/) и фактически является промышленным стандартом.

|  |  |  |
| --- | --- | --- |
| 1 | - | Резерв |
| 2 | CAN\_L | Линия шины CAN\_L (доминантная низкая) |
| 3 | CAN\_GND | Заземление CAN |
| 4 | - | Резерв |
| 5 | (CAN\_SHLD) | **Опционально**: экран CAN |
| 6 | (GND) | **Опционально**: заземление CAN |
| 7 | CAN\_H | Линия шины CAN\_H (доминантная высокая) |
| 8 | - | Резерв (линия ошибок) |
| 9 | CAN\_V+ | **Опционально**: питание |

*Для пользователей продукции KVASER*: Пожалуйста заметьте, что специфическое употребление этих контактов в кабелях KVASER DRVcan описано в документе LAPcan Hardware Guide, который можно скачать на сайте компании.

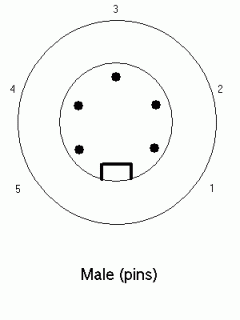
Если питание подается, оно должно быть в диапазоне +7..+13 В, 100 мA. Модули оснащены разъемом типа «папа» и должны соединять внутри контакты 3 и 6.

Нумерация контактов действительна для разъема типа «папа„, при взгляде со стороны разъема, или для разъема типа “мама», при взгляде со стороны распайки. – Чтобы запомнить расположение контактов, заметьте, что контакт CAN\_LOW имеет МЕНЬШИЙ (LOW) номер, а CAN\_HIGH – БОЛЬШИЙ (HIGH).



**5-контактный Mini–C**

Используется как [DeviceNet](http://odva.org/) , так и [SDS](http://www.honeywell.com/sensing/prodinfo/sds) , и является совместимым для этих двух протоколов.



|  |  |  |
| --- | --- | --- |
| Контакт | Функция | Цвет DeviceNet |
| 1 | Экран | Неизолированный |
| 2 | V+ | Красный |
| 3 | V- | Черный |
| 4 | CAN\_H | Белый |
| 5 | CAN\_L | Синий |

Модули оснащены разъемами типа «папа». Подаваемое напряжение 24 В ±1%

**6-контактный Deutsch DT04-6P**

Рекомендован CANHUG для использования в транспортных гидравлических системах

Разъемы на модулях типа «папа», разъемы шины – «мама». На данный момент нет никаких рекомендаций по вопросу подачи питания.

|  |  |  |
| --- | --- | --- |
| Контакт | Функция | Рекомендованный цвет кабеля |
| 1 | «Минус» питания | Черный |
| 2 | CAN\_H | Белый |
| 3 | **Опционально**: заземление сигнала | Желтый |
| 4 | **Опционально**: запуск | Серый |
| 5 | «Плюс» питания | Красный |
| 6 | CAN\_L | Синий |

**Тактовая синхронизация CAN**

**Схема бита**

Каждый бит, передаваемый по шине CAN, разделяется, для нужд тактовой синхронизации, как минимум на 4 части (кванта). Часть логически делится на 4 группы или сегмента:

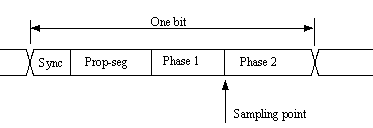
• сегмент синхронизации

• сегмент воспроизведения

• сегмент фазы 1

• сегмент фазы 2

*Схема бита данных шины CAN:*



Сегмент синхронизации, который всегда имеет длину в один квант, используется для синхронизации тактовых частот. Ожидается, что край бита появится здесь при смене данных на шине.

Сегмент воспроизведения нужен для компенсации задержки на линиях шины.

Сегменты фазы могут быть сокращены (сегмент фазы 1) или удлинены (сегмент фазы 2), если это потребуется для сохранения синхронизованности тактовых частот.

Уровни шины замеряются на границе между сегментом фазы 1 и сегментом фазы 2.

Большинство контроллеров CAN также обеспечивают возможность трехкратного замера на протяжении одного бита. В таком случае, замер происходит на границах двух квантов, предшествующих точке замера и результат зависит от мажоритарного декодирования (это верно как минимум в случае 82527).

**Тактовая синхронизация**

Для того, чтобы регулировать встроенный в чип генератор тактовых частот шины, контроллер CAN может сократить или удлинить бит на целое число квантов. Максимальное количество таких временных поправок бита определяется параметром «ширина скачка синхронизации» (Synchronization Jump Width, SJW).

**Жесткая синхронизация** происходит при переходе стартового бита от рецессивного к доминантному. Отсчет времени прохождения бита начинается заново с этой границы.

**Повторная синхронизация** происходит когда край бита не попадает в сегмент синхронизации сообщения. Один из сегментов фазы укорачивается или удлиняется на некоторое количество квантов, зависящее от ошибки фазы сигнала; максимальное количество используемых квантов определяется параметром «ширина скачка синхронизации» (Synchronization Jump Width, SJW).

**Вычисление регистра тактовой синхронизации**

Большинство контроллеров CAN позволяют программисту осуществлять настройку тактовой синхронизации используя следующие параметры:

• Значение предварительного делителя тактовой частоты

• Количество квантов перед точкой замера

• Количество квантов после точки замера

• Количество квантов в «ширина скачка синхронизации» (Synchronization Jump Width, SJW)

Обычно для этих целей выделяется два регистра: btr0 и btr1. Однако они могут слегка различаться у разных контроллеров, поэтому внимательно читайте инструкцию.

В контроллерах 82c200 и SJA1000, производства NXP (ранее Philips), раскладка регистра выглядит приблизительно так:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| btr0 | SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 |
| btr1 | SAM | TSEG22 | TSEG21 | TSEG20 | TSEG13 | TSEG12 | TSEG11 | TSEG10 |

• BRP0..BRP5 устанавливают значение предварительного делителя тактовой частоты

• SJW0..SJW1 устанавливают длину SJW

• TSEG10..TSEG13 устанавливают количество квантов перед точкой замера (стартовый бит не включен)

• TSEG20..TSEG22 устанавливают количество квантов после точки замера

• SAM при установке значения 1 производится три замера, при установке значения 0 – один замер

Примечание: реальные значения этих параметров несколько отличаются от значений, вписанных в регистр.

Пример: если сигнал генератора, подаваемый на SJA1000, имеет частоту 16 МГц, и мы желаем получить скорость передачи 250 кбит/с, с точкой замера в районе 62% всего бита, и SJW равным 2 квантам, мы можем установить –  
BRP = 4, что дает продолжительность кванта 2 × 4 / 16000000 с = 500 нс, и

TSEG1 = 5, что дает 5 квантов перед точкой замера, и

TSEG2 = 3, что дает 3 кванта после точки замера.

Каждый бит будет содержать 5 + 3 = 8 квантов, что даст нам желаемую скорость передачи 1 / (8 × 500 нс) = 250 кбит/с. Значения регистра должны быть следующими:

|  |  |
| --- | --- |
| btr0= | (SJW – 1) \* 64 + (BRP -1) = (2-1)\*64 + (4-1) = 67 = **0×43** |
| btr1= | SAM \* 128 + (TSEG2 – 1)\* 16 + (TSEG1 – 1) = 0×128 + (3-1)\*16 + (4-1) = («4» потому, что стартовый бит не включен) 35 = **0×23** |

Точка замера в районе 5/8 = 62.5% бита.

**Обработка ошибок CAN**

**Как CAN обрабатывает ошибки**

Обработка ошибок встроена в протокол CAN и очень важна для производительности системы CAN. Обработка ошибок нацелена на обнаружение ошибок в сообщениях, передающихся по шине CAN, чтобы передатчик мог повторно выслать неверно принятое сообщение. Каждый CAN–контроллер на шине будет пытаться обнаружить ошибку в сообщении. Если ошибка найдётся, обнаруживший её узел будет передавать флаг ошибки, таким образом разрушая трафик шины. Другие узлы обнаружат ошибку, вызванную флагом ошибки (если еще не обнаружили оригинальную ошибку) и предпримут соответствующие действия, т.е. отбракуют текущее сообщение.

Каждый узел обслуживается двумя счетчиками ошибок: счетчиком ошибок передачи (Transmit Error Counter) и счетчиком ошибок приёма (Receive Error Counter). Существуют правила, регламентирующие повышение и/или понижение значения этих счетчиков. По существу, передатчик определяет повышение числа сбоев в счетчике ошибок передачи быстрее, нежели слушающие узлы увеличат значения своих счетчиков ошибок передачи. Это потому, что есть немалая вероятность, что сбой именно в передатчике! Когда значение любого счетчика ошибок превышает определенную величину, узел сначала становится Error Passive – это значит, что он не будет активно разрушать трафик шины при обнаружении ошибки; а затем Bus Off – это значит, что узел вообще не будет принимать участия в передаче данных по шине.

При помощи счетчиков ошибок узел CAN может не только обнаруживать сбои, но и ограничивать ошибки.

**Механизмы обнаружения ошибок**

Протокол CAN описывает не менее пяти различных способов обнаружения ошибок. Два из них работают на уровне бита, а остальные три – на уровне сообщения.

1.Мониторинг битов (Bit Monitoring).

2.Вставка битов (Bit Stuffing).

3.Проверка кадра (Frame Check).

4.Проверка распознавания (Acknowledgement Check).

5.Проверка циклической избыточности (Cyclic Redundancy Check).

**Мониторинг бита**

Каждый передатчик шины CAN осуществляет мониторинг (т.е. повторное прочтение) переданного уровня сигнала. Если уровень прочитанного бита отличается от уровня переданного, подается сигнал ошибки бита (Bit Error). (Роста бита ошибок в процессе разрешения конфликтов не происходит.) Вставка битов

После того как узел передаст пять непрерывно следующих друг за другом битов одного уровня, он добавит к исходящему потоку битов шестой бит, противоположного уровня. Получатели будут удалять этот дополнительный бит. Это делается для предупреждения появления излишнего количества компонентов DC на шине, но также дает получателям дополнительную возможность обнаружения ошибок: если по шине передается более пяти непрерывно следующих друг за другом битов одного уровня, подается сигнал ошибки вставки.

**Проверка кадра**

Некоторые части сообщения CAN имеют фиксированный формат, т.е. стандарт четко определяет, какие уровни должны произойти и когда. (Эти части – ограничитель CRC (CRC Delimiter), ограничитель ACK (ACK Delimiter), конец кадра (End of Frame), а также пауза (Intermission), однако для них существуют дополнительные специализированные правила проверки на ошибки.) Если контроллер CAN обнаружит неверное значение в одном из этих полей, он подаст сигнал ошибки формы (Form Error).

**Проверка распознавания**

Ожидается, что все узлы шины, которые получили сообщение корректно (независимо от того, было ему это сообщение «интересно» или нет), отправят доминантный уровень в так называемой области распознавания (Acknowledgement Slot) кадра. Передатчик будет передавать рецессивный уровень. Если передатчик не сможет обнаружить доминантный уровень в области распознавания, он подаст сигнал ошибки распознавания (Acknowledgement Error).

**Проверка циклической избыточности**

Каждое сообщение содержит 15–битную контрольную сумму циклической избыточности (Cyclic Redundancy Checksum, CRC), и любой узел, обнаруживший что CRC в сообщении отличается от посчитанного им, подаст сигнал ошибки CRC (CRC Error).

**Механизмы ограничения ошибок**

Каждый контроллер CAN шины будет пытаться обнаружить описанные выше ошибки в каждом сообщении. Если ошибка обнаружится, нашедший её узел передаст флаг ошибки, таким образом разрушая передачу данных по шине. Другие узлы обнаружат ошибку, вызванную флагом ошибки (если они ещё не обнаружили оригинальную ошибку) и предпримут соответствующее действие, т.е. сбросят текущее сообщение.

Каждый узел обслуживают два счетчика ошибок: счетчик ошибок передачи и счетчик ошибок приема. Существуют правила, описывающие условия повышения и/или понижения значений этих счетчиков. По существу, передатчик, обнаруживший сбой, повышает значение своего счетчика ошибок передачи быстрее, чем слушающие узлы повысят значения своих счетчиков ошибок приема. Это потому, что есть большая вероятность, что сбоит сам передатчик!

Узел начинает работу в режиме Error Active. Когда значение любого из двух счетчиков ошибок превысит 127, узел перейдет в состояние Error Passive, а когда значение счетчика ошибок передачи превысит 255, узел перейдёт в состояние Bus Off.

• Узел в режиме Error Active при обнаружении ошибки будет передавать флаги активной ошибки (Active Error Flags).

• Узел в режиме Error Passive при обнаружении ошибки будет передавать флаги пассивной ошибки (Passive Error Flags).

• Узел в режиме Bus Off не будет передавать ничего.

Правила повышения и понижения значений счетчиков ошибок довольно сложные, но принцип прост: ошибка передачи добавляет 8 пунктов, а ошибка прием – 1 пункт. Правильно переданные и/или принятые сообщения вызывают понижение значения счетчика(ов).

Пример (слегка упрощенный): Представим, что у узла A плохой день. Всякий раз, когда A пытается передать сообщение, происходит сбой (не важно, по какой причине). При каждом сбое значение счетчика ошибок передач увеличивается на 8 пунктов и передается флаг активной ошибки. Затем он пытается послать сообщение ещё раз.. и всё повторяется.

Когда значение счетчика ошибок передачи превысит 127 пунктов (т.е. после 16 попыток), узел A перейдёт в режим Error Passive. Разница в том, что теперь он будет передавать флаги пассивной ошибки. Флаг пассивной ошибки содержит 6 рецессивных битов и не будет нарушать передачу других данных по шине – поэтому другие узлы не услышат жалобы A на ошибки шины. Однако A продолжит повышать значение счетчика ошибок передачи. Когда он превысит 255 пунктов, узел A окончательно сдастся и перейдет в режим Bus Off.

Что другие узлы думают об узле A? – После каждого флага активной ошибки, переданного узлом A, остальные узлы повышают значения своих счетчиков пассивной ошибки на 1 пункт. За всё то время, что потребуется узлу A для перехода в режим Bus Off, значения счетчиков ошибок получения остальных узлов не превысят границы Error Passive, т.е. 127. Это значение будет уменьшаться на 1 пункт при каждом корректном получении сообщения. Однако узел А будет оставаться в режиме Bus Off.

Большинство контроллеров CAN будут предоставлять биты статуса (и соответствующие прерывания) для двух состояний:

• «Предупреждение об ошибке» (Error Warning) – значение одного или обеих счетчиков ошибок превысило 96 пунктов

• Bus Off, как описано выше.

Некотрые, но не все (!), контроллеры также предоставляют бит для состояния Error Passive. Немногие контроллеры также предоставляют прямой доступ к счетчикам ошибок.

Привычка контроллеров CAN автоматически переотправлять сообщения при возникновении ошибок иногда может раздражать. На рынке имеется как минимум один контроллер (SJA1000 от Philips), поддерживающий полное ручное управление обработкой ошибок.

**Режимы сбоев шины**

Стандарт ISO 11898 перечисляет несколько режимов сбоев кабеля шины CAN:

1.CAN\_H прерван

2.CAN\_L прерван

3.CAN\_H короткозамкнутый на напряжение батаре

4.CAN\_L короткозамкнутый на землю

5.CAN\_H короткозамкнутый на землю

6.CAN\_L короткозамкнутый на напряжение батареи

7.CAN\_L короткозамкнутый на провод

8.CAN\_H и CAN\_L прерваны в одном и том же месте

9.Потеря соединения с оконечной нагрузкой сети

Для сбоев 1–6 и 9 «рекомендовано», чтобы шина сохраняла работоспособность путём снижения соотношения сигнал/шум (S/N), а в случае сбоя 8 – чтобы исходная подсистема сохранила работоспособность. Для сбоя 7 существует «опциональная» возможность сохранения работоспособности путём снижения соотношения сигнал/шум (S/N).

На практике система CAN, построенная на приемопередатчиках типа 82C250, не сохранит работоспособность при сбоях 1–7, а при сбоях 8–9 может как сохранить, так и не сохранить.

Существуют «устойчивые к сбоям» драйверы, такие как TJA1053, способные обрабатывать все сбои. Обычно за эту устойчивость приходится платить ограничением максимальной скорости; для TJA1053 она составляет 125 кбит/с.

class BluetoothSocket:

\_\_doc\_\_ = \_bt.btsocket.\_\_doc\_\_

def \_\_init\_\_ (self, proto = RFCOMM, \_sock=None):

if \_sock is None:

\_sock = \_bt.btsocket (proto)

self.\_sock = \_sock

self.\_proto = proto

Библиотека pybluez предоставляет интерфейс для работы с Bluetooth в Python, и в данном случае используется для соединения с устройством по протоколу RFCOMM — аналогом протокола последовательного порта (Serial Port Protocol), оптимизированным для Bluetooth.

Вот что происходит на каждом этапе выполнения вашего кода:

**1. socket = bluetooth.BluetoothSocket(bluetooth.RFCOMM)**

В этом шаге создается объект сокета, который будет использоваться для общения с устройством по Bluetooth:

* **bluetooth.BluetoothSocket** — это класс из библиотеки pybluez, представляющий Bluetooth-сокет. Он похож на обычный сокет, используемый в сетевом программировании, но оптимизирован для Bluetooth-коммуникации.
* **bluetooth.RFCOMM** — константа, указывающая на тип сокета, который нужно создать. Протокол RFCOMM (Radio Frequency Communication) предназначен для эмуляции последовательного порта через Bluetooth. Он надежен, поддерживает подключение "точка-точка" и работает поверх L2CAP (Logically Link Control and Adaptation Protocol).

Важно: RFCOMM используется для того, чтобы передавать данные небольшими пакетами, и часто применяется при подключении к устройствам с последовательным интерфейсом, например, сканерам OBD-II, медицинскому оборудованию, клавиатурам и другим периферийным устройствам.

Создание объекта BluetoothSocket на этом этапе фактически инициализирует Bluetooth-сокет на уровне операционной системы, но само подключение еще не выполняется.

**2. socket.connect((OBD2\_MAC\_ADDRESS, OBD2\_PORT))**

Этот метод инициирует подключение к Bluetooth-устройству, используя заданный адрес и порт.

* **socket.connect()** — метод, аналогичный методу connect() для обычного сокета. Он используется для установки соединения с другим Bluetooth-устройством по указанному адресу и порту.
* **(OBD2\_MAC\_ADDRESS, OBD2\_PORT)** — кортеж, содержащий:
  + **OBD2\_MAC\_ADDRESS** — строку с MAC-адресом устройства, с которым необходимо соединиться. Этот MAC-адрес уникально идентифицирует Bluetooth-устройство.
  + **OBD2\_PORT** — номер порта на устройстве, к которому выполняется подключение. В случае с OBD-II адаптерами обычно используется порт 1, так как многие Bluetooth-устройства используют этот порт по умолчанию для RFCOMM-соединений.

**Что происходит "под капотом":**

* **Инициация соединения**. Когда вызывается connect, Bluetooth-сокет отправляет команду операционной системе на установку связи с указанным устройством. Операционная система инициирует процедуру "пейринга" (если устройства еще не были сопряжены), или сразу начинает процедуру подключения.
* **Соединение на уровне RFCOMM**. Если устройство ответило на запрос на соединение, создается канал RFCOMM, который позволяет устройствам передавать данные.

Если устройства не сопряжены, и это не было выполнено заранее, этот шаг может потребовать авторизации (запроса на пейринг). Также могут появиться ошибки, если устройство недоступно, адрес неверен, или используется неправильный порт.

**Пример использования и обработки ошибок**

На практике важно обрабатывать возможные ошибки соединения, так как Bluetooth — нестабильная среда, и сбои могут возникнуть по разным причинам (например, устройство может быть вне зоны действия или занято).