Malta Police Training

The automotive CAN bus

1. What is CAN Bus?

- The **CAN bus** is a communication system used in vehicles to allow different electronic components to communicate with each other.
- It's a **two-wire** network: CAN High (CAN_H) and CAN Low (CAN_L), allowing devices (ECUs—Electronic Control Units) to send and receive messages.
- This system replaces older point-to-point wiring systems, reducing the amount of wiring needed and improving reliability.

2. Key Features

- **High-Speed Communication:** CAN operates at speeds up to 1 Mbps, which is faster than many other vehicle communication networks.
- **Real-Time Data:** CAN bus operates in real-time, meaning that information can be transmitted between ECUs quickly and efficiently.
- **Multi-Device Network:** Multiple devices can share the same bus, making it flexible and scalable for modern vehicles with complex systems.

3. Applications in Vehicles

- **Engine Control Unit (ECU):** CAN bus allows communication between the engine control unit, transmission control, fuel system, and other key components.
- **Safety Systems:** It's used for ABS, airbag systems, and other safety-critical systems.
- **Infotainment and Comfort:** Controls for lighting, infotainment, heating, and air conditioning can also communicate over the CAN bus.
- **Diagnostics:** Mechanic tools and diagnostic scanners (like OBD-II) use the CAN bus to read and troubleshoot vehicle problems.

4. Diagnosing and Troubleshooting

- **Scan Tool Access:** Mechanics use OBD-II scanners to communicate with the vehicle's ECUs through the CAN bus.
- **Checking for Faults:** If there's an issue with any component, the scanner can read trouble codes, which helps pinpoint faulty sensors or ECUs.
- **Signal Interruption:** A broken or disconnected wire in the CAN bus can cause communication errors between ECUs, potentially disabling certain vehicle systems.

5. CAN Protocol

- **Message Format:** CAN messages consist of an identifier (which determines the priority of the message), data, and error checking.

- **Error Detection:** It includes features like checksums, cyclic redundancy checks (CRC), and acknowledgments to ensure data integrity and correct transmission.

6. Types of CAN Buses in Vehicles

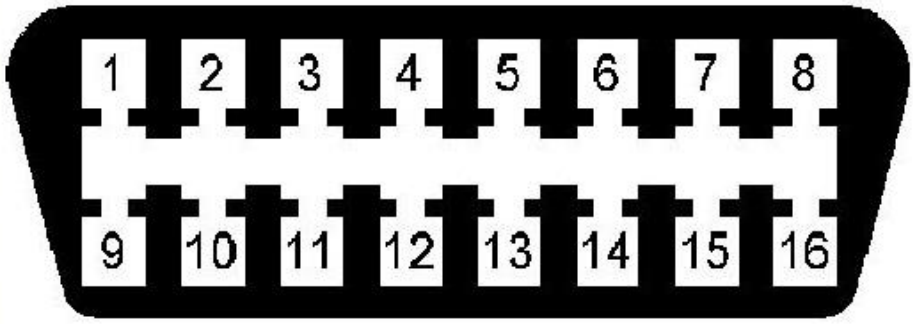
- **Standard CAN:** Uses an 11-bit identifier.
- **Extended CAN:** Uses a 29-bit identifier for more complex systems.

7. Tools and Equipment for Mechanics

- **CAN Bus Analyzer/Scanner:** Used to monitor CAN traffic and diagnose issues with the bus or individual components.
- **Oscilloscope:** In some cases, mechanics might use an oscilloscope to examine the electrical signal on the CAN bus.
- **Wiring Diagrams:** Having a clear wiring diagram specific to the vehicle can help troubleshoot the CAN bus.

8. Practical Considerations for Mechanics

- **Not All Systems are on the CAN Bus:** Some components might use other communication protocols, so it's important to know the system you're working on.



PIN	DESCRIPTION	PIN	DESCRIPTION
1	Vendor Option	9	Vendor Option
2	J1850 Bus +	10	j1850 BUS
3	Vendor Option	11	Vendor Option
4	Chassis Ground	12	Vendor Option
5	Signal Ground	13	Vendor Option
6	CAN (J-2234) High	14	CAN (J-2234) Low
7	ISO 9141-2 K-Line	15	ISO 9141-2 Low
8	Vendor Option	16	Battery Power

OBD-II Connector and Pinout

Literature Review

CAN Bus

Controller Area Network (CAN) is a specialised linear bus system designed by Robert Bosch GmbH for implementation in automotive applications. It is a means by which all nodes active on the bus (sensors and engine control units/ECUs) can communicate to one another bilaterally to relay relevant information essential for the vehicle to gain knowledge of the specific driving conditions at hand. It is an industry accepted protocol due to its low cost to implement, simplistic physical layer, signal error detection capabilities and automatic message retransmission functionality. Data is received and transmitted on this bus serially, over a two-wire differential bus comprising of the CAN-high and CAN-low signals. [4]

Topology

Each node must utilise a CAN controller chip which communicates on the bus via a transceiver. The transceiver translates the electrical representation of a bit as defined by the controller to that defined by the bus. When transmitting, it supplies adequate current output to ensure the bus maintains the required electrical state, whilst when receiving it determines the recessive signal level and safeguards overloading the CAN controller chip from excessive voltages present on the bus. The transceiver's role is also to detect operational errors within the CAN-bus such as short circuits, shorts to ground or line malfunctions. Figure 1 depicts a specific model of CAN-bus transceiver [5]. The CAN microcontroller's Tx connection is responsible for the data transmitting signal and is outputted from the MCU (micro-controller unit) whilst the Rx connection accepts the receiving signal and is an input to the MCU [6].

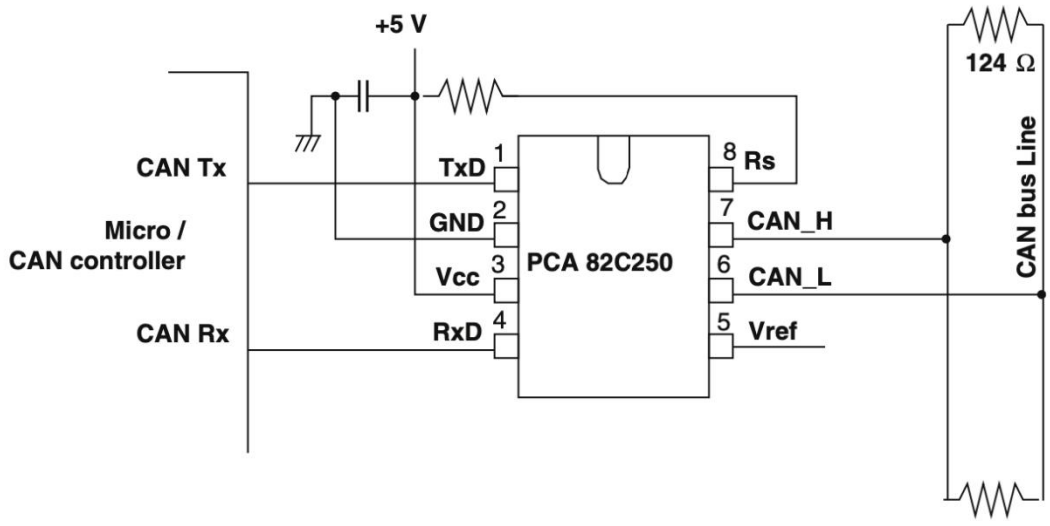


Figure 1 Schematic diagram of CAN-bus transceiver of model NXP PCA 82C250. [5]

The transceiver is connected to the bus via the CAN-high and CAN-low lines. Both these lines are terminated at the end of the bus by means of termination resistors which should equal the impedance of the cable. ISO 11898 (high speed CAN-bus standardisation) specifies cables with an impedance of 120Ω, thus termination resistors of around 120Ω must be used, as shown in Figure 1. This is performed so as to minimise signal reflections at the end of the line arising from imperfections in the cable, leading to conflicting impedances and non-linear variations to the characteristics of the cable [5].

Message Encoding

Messages on the CAN-bus are comprised of bits occupying one of two possible states, either dominant (logic 0) or recessive (logic 1). Figure 2 depicts the differential voltages required to encode both states. For a bit to switch states from recessive to dominant, the CAN-high and CAN-low signals must have a differential voltage greater than a certain threshold (0.5V for receiving data and 1.5V for data transmission). Due to this encoding, messages of higher priority acquire access to the CAN-bus first. This is configured in such a way so as to allow even a single node to change the status of the entire bus to 'dominant', whilst all other nodes are concurrently recessive [5]. This strategy is the defining characteristic by which message arbitration is enabled.

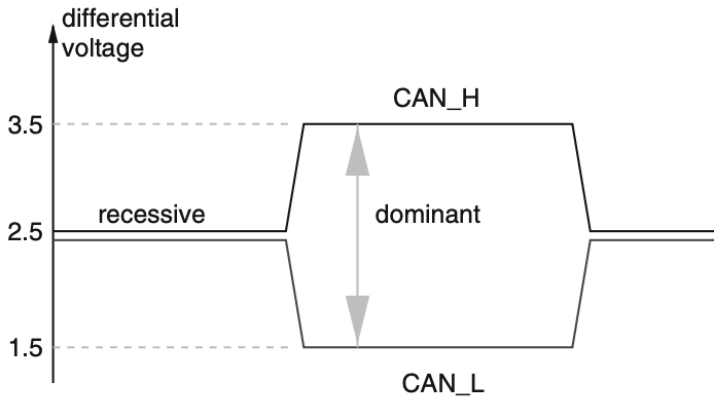


Figure 2: Encoding of dominant and recessive states. [5]

Message Structure

The CAN protocol stipulates 4 frames indicating the type of message capable of being communicated over CAN: data frames, remote frames, error frames and overload frames. For the scope of these notes, data frames and remote frames and only some specific frame elements are of the most interest to be analysed. Data frames are message types transmitted from a source node to relay information to other receiver

nodes. This type of message is usually transmitted from various sensors on the CAN-bus carrying information regarding certain parameters pertaining to relevant ECUs/nodes. The concerned nodes (ECUs) would know whether or not to receive this information according to the message's content encoded in the identifier field of the frame, thus filtering out any messages whose identifiers are not of interest to the node [5].

Figure 3 depicts the sequence of frame elements within a standard data frame, whilst Table 1 describes the function and length of each element within the data frame. Two types of data frames can co-exist within the same CAN-bus: standard and extended. Extended data frames were developed as part of the more updated CAN2.0 protocol. These data frames utilise a 29-bit identifier instead of the standard frame's 11-bit identifier. As both these types of data frames can be present on the CAN-bus simultaneously, arbitration is unaffected with extended data frames and is functional regardless of the identifier length.

The start-of-frame (SOF) bit denotes the start of a new CAN frame and allows all nodes to synchronise prior to a new message being sent. It also prompts each node to start comparing identifier bits in order to deduce which node has higher priority and should gain access first. The Remote-Transmission-Request (RTR) bit is used by destination nodes to request information from source nodes. If recessive, the message is a remote frame (requesting information), if dominant, a data frame (containing information). The DLC field (data-length-content) denotes the amount of bytes containing information in the message, which can range from 0 to 8. For remote frames it denotes the length of information requested by a destination node from a source node.

SOF	Identifier	RTR	IDE	r0	DLC	Data (0-8 Bytes)	CRC	ACK	EOF	IFS

Figure 3: Standard data frame. [6]

		transmission.
Identifier	11	An identifier for the data which also represents the message priority.
Remote Transmission Request (RTR)	1	Dominant. See Remote Frame subsection.
Identifier Extension (IDE)	1	Sets standard (0) or extended format (1).
Reserved bit (r0)	1	Reserved bit.
Data Length Code (DLC)	4	Number of bytes of data (0–8 bytes).
Data	0-64 (0-8 bytes)	Data to be transmitted.
CRC + delimiter	16	Cyclic Redundancy Check.
ACK slot + delimiter	2	Transmitter sends recessive and receiver asserts dominant on successful reception.
End Of Frame (EOF)	7	Marks the end of a CAN frame.
Interframe Space (IFS)	7	Contains the time required by the controller to move a correctly received frame to its proper position in a message buffer area.

Message arbitration

Arbitration is the method by which CAN systems avoid data collisions when multiple nodes attempt to communicate simultaneously. When a node wishes to communicate on the CAN-bus and finds that another node is already communicating, it waits for the end of the current bit to initiate an SOF bit. As the CAN protocol requires all nodes to synchronise on bit edges, transmission sequences which maintain the same state for a sustained period must be avoided. To maintain synchronisation whilst messages are being sent on the CAN-bus, a technique called ‘bit stuffing’ takes place. If a transmission sequence remains in the same state for a period of longer than 5 bits, a complemented bit is sent on the bus to ensure each node’s bit clock maintains synchronisation. This complemented bit is then decoded by the MCU of each node to read the original message.

Since a dominant bit (logic 0) would dominate the entire CAN-bus over other recessive bits (logic 1), lower message identifiers have higher priority over the bus. Each node must read the current bus state whilst themselves are attempting to transmit. If a node detects that each bit transmitted on the medium is identical to the identifier bits of said node, it realises it has the highest priority and it gains access to the bus to continue its message. The other nodes realise they have lost contention and switch to a read-only mode. The CAN protocol requires that all nodes wishing to

communicate on the same medium, must transmit messages with unique identifiers, allowing for arbitration to take place. This property is called ‘unique purity’. [5]

Nodes whose access to the bus was not initially granted due to their lower priority, retry transmitting the message 6 bit clocks after the higher priority message is finished and can fight for contention once more, until each message, cascading in priority, finally gains access to the bus. Figure 4 shows an example of 3 nodes requiring to communicate simultaneously on the CAN-bus. The binary encoding for each message identifier is shown under the hexadecimal ID of each node. Node 2 has the largest identifier, which implies that more higher power bits are recessive rather than dominant compared to the identifiers of the other 2 nodes. After all nodes synchronise using the start of frame bit, in this example all nodes send a dominant, logic 0 bit. Up till this point no node has conceded, however, the second bit outputted by node 2 is recessive, whilst the bits sent by nodes 1 and 3 are dominant and thus node 2 withdraws from contention, due to the voltage on the bus being in a dominant state. [5] [6]

Node 3 and Node 1 then both send a dominant bit and thus are still in competition, however, with the bit thereafter, node 3 withdraws due to its recessive bit and node 1 wins the competition and is allowed to send the remainder of its message whilst all other nodes remain in a listening mode. As each identifier is unique to each message present on the CAN-bus for any given moment in time, no two messages can have the same priority to access the CAN-bus simultaneously. Thus messages of higher importance, usually being communicated by nodes related to safety critical systems for example, have lower value CAN IDs and thus greater access to the bus.

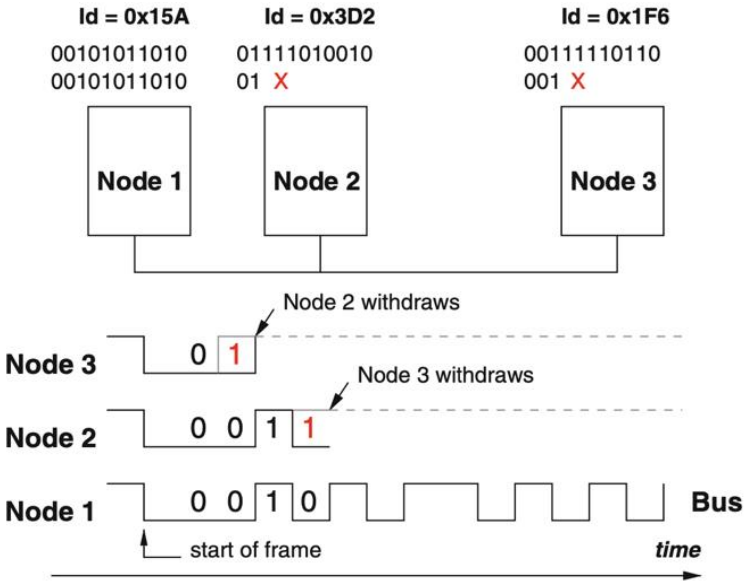


Figure 4: CAN-bus arbitration on message identifiers. [5]

Can bridges

CAN communication may be split over several buses, where each may be dedicated to separate vehicle functions. This is done so as to reduce the load on a single bus and mitigate message latency whilst maintaining the same, simplistic CAN protocol. When multiple CAN networks are present, wherein each may require to access information found only on the other bus, CAN bridges are utilised to allow communication between said networks, so long as both networks are based on the same protocol. Bridges are not to be confused with Gateways, which allow for communication between different protocols (CAN and FlexRay for example). [7] [8]

If one CAN network were to infrequently require information from a specific node on another bus, instead of both buses sharing said node, causing it to unnecessarily occupy bandwidth, a bridge may retrieve this information from the node's primary bus and relay it to the network inquiring for this information. Utilising bridges, usual CAN limitations (signal reflections, line inductance mismatches etc.) that arise from increasing bus length or increasing volume of nodes can be resolved due to the reduction in bus traffic and segmentation of bus lines. [7]

Bridges are differentiated from repeaters, as repeaters reproduce the exact same electrical signals from one network to another. If two networks were to be physically divided however interlinked using a repeater, they would logically be one bus. With bridges, these two networks would be logically separate due to their store-and-forward mode of operation. This mechanism implies that the node requiring the data located in the other bus receives the message (or portions thereof) after a short period of time, as the message is retransmitted such that the receiving node can interpret it. This is performed due to the bridge's capability of recognising active nodes within the bus and generating address tables to control message routing. [5] [7]

1.1 Other Communication Protocols

Over the last 30 years, the CAN communication protocol has become one of the most implemented communication protocols in the automotive industry. However, with the highly competitive automotive market embracing advancements into new technologies, such a protocol is starting to show its age. CAN's technical philosophy based on its physical topology limit it to a bit rate of $1 Mbits^{-1}$, any faster bit rates as dictated by advancing automotive technologies, with such an architecture, would require for the protocol to work around physical line phenomena such as transmission reflections. [9]

Another limitation of CAN is the 'event-triggered' philosophy of the protocol, wherein messages are independent of time, however reliant on parametric changes to the system for communication to initiate (ECUs request information on command or sensors transmit data at a rate dependent of their function). The entire arbitration process is another drawback of CAN, wherein nodes must wait for higher priority nodes to communicate prior to gaining access to the bus. This leads to all messages other than the one with the highest priority to have little control over when a message is sent, albeit such a message still has to wait for the arbitration process to complete for it to transmit. Due to the sheer volume of messages present on a CAN-bus in a modern day vehicle, the CAN protocol would find it immensely difficult to allow for communication redundancies to occupy bandwidth on the bus. As mentioned prior, such a problem can be alleviated by means of implementing multiple CAN-buses

dedicated to different functional groups so as to reduce bus traffic. However, this is merely a temporary solution to an approaching obstacle. [5] [9]

More modern protocols must work around these issues to allow technologies such as ‘all-by-wire’ (X-by-wire) to be implemented. This would allow for the removal of mechanical interfaces between certain safety-critical vehicle controls such as the steering or brake controls as found in the commercial aviation industry (fly-by-wire). Such systems are however being designed with safety measures in mind which cannot be distilled to a level of simplicity that the CAN protocol can accommodate.

Time-Triggered CAN (TTCAN)

The aim of TTCAN is to increase the predictability of CAN by specifying a message’s latency irrelevant of the load on the bus, hence giving CAN a more time-dependent philosophy. TTCAN operates by dictating a basic operating cycle within which each node is allowed to communicate depending on the rate at which each node requires access to the bus. Figure 5 depicts the standard operating cycle for TTCAN, wherein each row represents a single cycle which loops after a specific period of time, whilst every column depicts the time interval allocated to each node to transmit their relevant messages. [9]

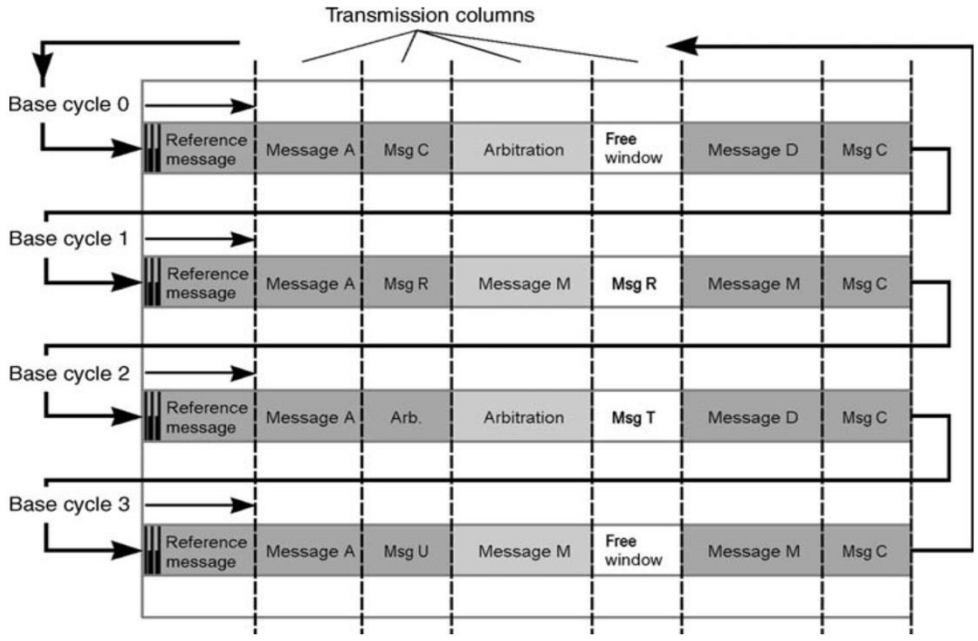


Figure 5: TTCAN's operating cycle depicted as a table of rows and columns. [9]

Such an operating principle requires each node to maintain synchronisation within the cycle such that it does not impede the node after it. This is supported by a ‘time master’ node which acts as a frame synchronisation entity, communicating to all other nodes the time allocation within which each is authorised to communicate [10]. This allocation is altered by the time master depending on the changing requirements of each node within the bus. There are specified periods within the cycle which allows all nodes to communicate subject to CAN arbitration, mimicking the original CAN

protocol. Due to the inherent structure of a CAN message, and the inconsistent time lost during message arbitration, TTCAN does not equate to a real-time system. However, if the messaging rate of each cycle is fast enough, allowing the same node to communicate repeatedly within a short frame of time, each node is provided with ‘quasi-real time’ access to the bus. [9]

Flexray

FlexRay was developed in 2004 by the FlexRay consortium, comprising of major automotive and technological corporations such as BMW, DaimlerChrysler, General Motors, Bosch, Philips, and others, aiming to create a communications network capable of accommodating automotive systems within which CAN was deficient. FlexRay inhibits message collisions on the transmission medium which in turn allows nodes access to the bus without message arbitration (specifically for the static part of a FlexRay communication cycle, mentioned later). The exclusion of the arbitration process constitutes real-time communication, where each node is allowed to transmit at a defined instant in time, for a known duration, with the certainty that no other node requires bus access at this given instant [11]. FlexRay, similar to TTCAN, utilises repeating communication cycles wherein nodes are allowed to communicate during specified time slots. Unlike TTCAN, which utilises a time master node to allocate these time slots to other nodes, FlexRay’s time allotment for each node (ECU) is chosen by the designer (manufacturer) during the network’s design phase, however, there is still a node present tasked with being the network manager.

To increase FlexRay’s applicability, the protocol incorporates both a time-triggered and an event-triggered philosophy by means of segmenting a standard communication cycle into both a static part and a dynamic part. Operating respectively by means of fixed Time Division Multiple Access (TDMA) and flexible TDMA. Figure 6 depicts varying configurations of a FlexRay communication cycle, such that within the static part, real time communication occurs, whilst in the dynamic part, network messaging is permitted at a variable bit rate, necessitating the need for an unspecified communication time and requiring communication to undergo arbitration.

Both parts comprising a cycle are further segmented into mini time slots, however, the difference being that a node may freely vary the duration of its message in the dynamic part. This network design was never intended to replace CAN, due to the indispensability and presence of CAN within the market, however it was more aimed at supporting high-speed technologies such as X-by-wire to be implemented without needing to reengineer other tried-and-tested systems. [9] [11]

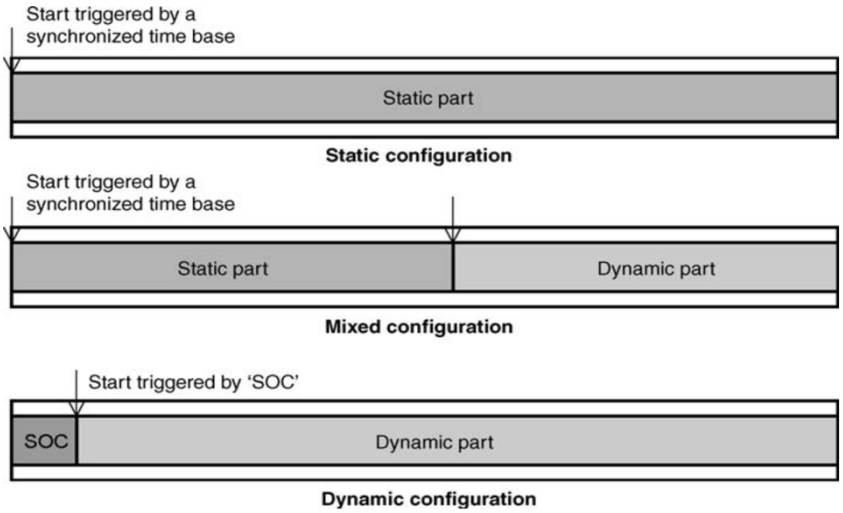


Figure 6: Different communication cycle configurations. [9]

Gateway

With the labyrinth of electronic systems now present in modern day vehicles, none of these aforementioned protocols are being favoured over the other, nevertheless, all are being implemented in conjunction to one another so as to tackle different subsystems within the vehicle according to each protocol's strengths. To allow communication between networks utilising different higher layer protocols or different bitrates, gateways are implemented to translate messages between networks and support communication between different systems within the vehicle [5].

Figure 7 depicts two CAN-controlled subsystems (powertrain and body/chassis) being interlinked using a gateway. Due to the different bitrates, messages on one bus would be illegible on the other, and by utilising a gateway, communication between the two buses can occur. For this example, such a gateway would allow for the vehicle to lock the doors when a certain speed is reached. The Body-bus (B-bus) would transmit a remote frame, inquiring about the vehicle, where the gateway retransmits the message to the Powertrain-bus (PT-bus).

The reply is then identified by the gateway on the PT-bus and conveyed to the B-bus, allowing the door module to lock the doors if necessary [8]. Such functionality allows manufacturers to implement different protocols within the same vehicle according to a system's networking needs. The Volvo XC90 allows communication between 40 ECUs dispersed between low-speed CAN, high-speed CAN, Local Interconnect Network (LIN) and Media-Oriented System Transport (MOST) buses through the use of gateways. [5] [8] [12]

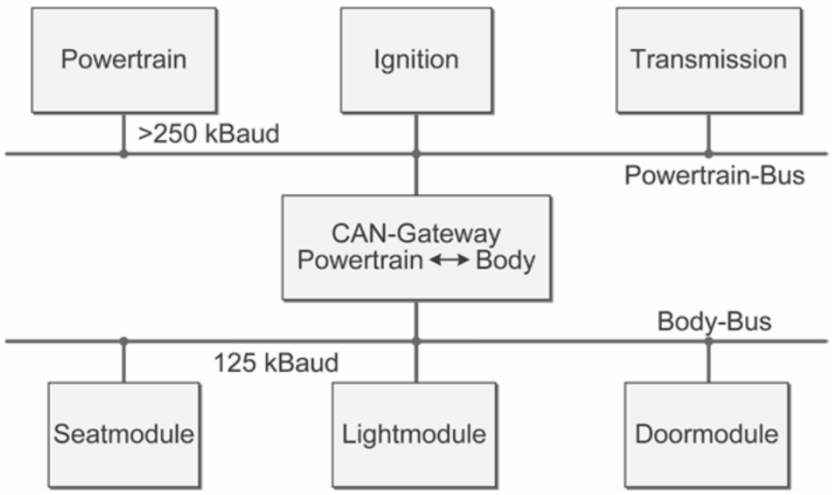


Figure 7: Two CAN-buses operating at different bitrates allowed to communicate through a gateway. [8]

Literature Review above from Thor Scicluna B.Eng. Dissertation 2023 "Investigating Euro 6 Diesel Vehicle Emissions After-treatment using Chassis Dynamometer and CAN-bus data".

[4] Robert Bosch GmbH, Diesel engine management: Systems and components, vol. 1, K. Reif, Ed., Wiesbaden, Hasse: Springer Vieweg, 2014.

[5] M. Di Natale, H. Zeng, P. Giusto and A. Ghosal, Understanding and using the Controller Area Network Communication protocol, New York: Springer, 2012.

[6] Future Technology Devices International Limited, "What is CAN?," FTDI Chip, Glasgow, 2015.

[7] H. Ekiz, A. Kutlu and E. Powner, "Design and implementation of a CAN/CAN bridge," Proceedings Second International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'96), pp. 507-513, 14 June 1996.

[8] W. Lawrenz, Ed., CAN System Engineering: From Theory to Practical Applications, Wolfenbüttel: Springer-Verlag London, 2013.

[9] D. Paret, Multiplexed Networks for Embedded Systems; CAN, LIN, Flexray, Safeby-Wire..., Chichester, West Sussex: John Wiley & Sons Ltd, 2007.

[10] The International Organization for Standardization, "Road vehicles — Controller area network (CAN) — Part 4: Time-triggered communication," in ISO 11898-4:2004, 2004.

[11] D. Paret, FlexRay and its Applications: Real time multiplexed network, Hoboken, NJ:Wiley, 2012.

OBD-II Parameter Identifiers , PIDs, source wikipedia

OBD-II PIDs (On-board diagnostics Parameter IDs) are codes used to request data from a vehicle, used as a diagnostic tool.

Services / Modes

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	Show current data
02	Show freeze frame data
03	Show stored Diagnostic Trouble Codes
04	Clear Diagnostic Trouble Codes and stored values
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	Request vehicle information
0A	Permanent Diagnostic Trouble Codes (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 (HEV).[2]

Standard PIDs

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).

A								B								C								D							
A 7	A 6	A 5	A 4	A 3	A 2	A 1	A 0	B 7	B 6	B 5	B 4	B 3	B 2	B 1	B 0	C 7	C 6	C 5	C 4	C 3	C 2	C 1	C 0	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0

Service 01 - Show current data

]

PIDs (hex)	PID (Dec)	Data bytes returned	Description	Min value	Max value	Units	Formula[a]
00	0	4	PIDs supported [$01 - $20]				Bit encoded [A7..D0] == [PID $01..PID $20] See below
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
02	2	2	DTC that caused freeze frame to be stored.				Decoded as in service 3
03	3	2	Fuel system status				Bit encoded. See below

PIDs (hex)	PID (Dec)	Data bytes returned	Description	Min value	Max value	Units	Formula[a]
04	4	1	Calculated engine load	0	100	%	(or)
05	5	1	Engine coolant temperature	-40	215	°C	
06	6	1	Short term fuel trim (STFT)—Bank 1	-100 (Reduce Fuel: Too Rich)	99.2 (Add Fuel: Too Lean)	%	(or)
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)	0	765	kPa	
0B	11	1	Intake manifold absolute pressure	0	255	kPa	
0C	12	2	Engine speed	0	16,383.75	rpm	
0D	13	1	Vehicle speed	0	255	km/h	
0E	14	1	Timing advance	-64	63.5	° before TDC	
0F	15	1	Intake air temperature	-40	215	°C	

PIDs (hex)	PID (Dec)	Data bytes returned	Description	Min value	Max value	Units	Formula[a]
10	16	2	Mass air flow sensor (MAF) air flow rate	0	655.35	g/s	
11	17	1	Throttle position	0	100	%	

From OBD-II Parameter Identifiers , PIDs, till here, source is wikipedia

Non Standard PIDs,

Non standard PID can be found by first using an Automotive Scan Tool to find the desired parameter, eg the Soot Loading on the DPF of a Renault DCi engine. While the scan tool is showing the Soot Loading value, a sniffer tool is connected on a splitter cable to overhear the Scan Tool information. It is noted that a splitter means a cable that from one OBD port provides multiple OBD ports so that a tee junction is available such that multiple CAN bus modules (in this case the Automotive Scan tool and the sniffer) can be simultaneously connected to the vehicle.

The sniffer software has to be able to filter out all messages and leave only those pertaining to the Scan tool question and the respective vehicle answer to that request. This is most effectively done by first connecting the sniffer to the vehicle CAN bus without the scan tool. The messages that are overheard are then filtered out by applying a filter in the sniffer that avoids showing all the messages that occur on the bus. Then the scan tool is connected and the messages that come up will be those pertaining to the scan tool and the vehicle response to the scan tool. Then the desired parameter, example the Soot loading, is requested and the vehicle responds. The scan tool question and the vehicle response can thus be seen. The translation of the data in the response to a meaningful value in decimal is more complicated because a means to alter the value needs to be found so that the scan tool can report the variation in the parameter. This can be done by having yet another computer that can send out messages on the CAN bus. This computer is first commanded to send exactly the same reply as the vehicle, then the vehicle CAN bus is disconnected (example by switching a relay) and the computer can then reply instead of the vehicle. The data sent out as a reply is then changed by trial and error to see what data means what decimal value of the parameter and a calibration table (or actually equation)is built. This methodology has been used by the author of these notes on a handful of vehicles. It is time consuming but can be done.

Sniffing of data between ECUs

It is also good to know that data can be obtained by overhearing rather than by question and answer. As an example, the ABS module really has the wheel speed information because the wheel speed sensors are connected to the ABS module. The ABS module broadcasts the four wheel speed values for use by other ECU's example the display which needs to be displaying the speedo. Likewise the engine ECU has the sensors that are able to generate the engine RPM, hence the engine ECU broadcasts the engine RPM so that display can show the engine RPM to the driver. These values are broadcast without a request being sent, which is different to what happens with the SCAN tool which asks a parameter and a reply is sent back as a response to that question. The broadcast information is typically sent out very frequently and hence can be a source of very fast information without any need to install sensors.

This type of data capture by sniffing was performed on three different vehicles by the author of these and for example on an Auris was done to capture relevant data and compare it with the data recorded by the Crash Data Recorder.

Table of the relevant CAN ID of the Toyota Auris 2007 and the equation that translates the CAN data to data in the stated units.

CAN ID (HEX)	CAN Message Length (Bytes)	Rate (*ms*)	Parameter Conveyed	Translational Equation	Units
00B4	8	20	Vehicle Speed	$\frac{(D5 \times 256) + D6}{100}$	*km/h*
610	8	500	Vehicle Speed	$D2$	*km/h*
3B4	8	1000[*1]	Brake (Stop Light Switch)	$D4$	1/0
3B3	3	500	Engine Speed	$\frac{(D0 \times 256) + D1}{1.28}$	RPM
2C4	8	20	Engine Speed	$\frac{(D0 \times 256) + D1}{1.28}$	*RPM*
2C1	8	30	Engine Speed	/	*RPM*
3B3	3	500	Accelerator Pedal Position (APP)	Using Eq. [6, 7]: $y_D = 0.8856x_C - 69.5470$, and $y_D = 45x_L - 67.1$	*% and V*
00B0	6	10	Front Right Wheel Speed	$\frac{(D0 \times 256) + D1}{100}$	*km/h*
		10	Front Left Wheel Speed	$\frac{(D2 \times 256) + D3}{100}$	*km/h*
00B2	6	10	Rear Right Wheel Speed	$\frac{(D0 \times 256) + D1}{100}$	*km/h*
		10	Rear Left Wheel Speed	$\frac{(D2 \times 256) + D3}{100}$	*km/h*

[*1] For the Brake Status change on CAN ID 3B4, it should be noted that if there is no change in brake status, the refresh rate is every 1 *s*. However, if there is a change of state of the brake, the data is refreshed immediately.

Commercially available CAN bus analyser and sniffer is the following from National Instruments which we also have t University and found it good. It can be used with stand alone software that I believe is free Xnet and also can be used with LabVIEW which however is not free.

NI USB 8502

https://www.ni.com/it-it/shop/model/usb-8502.html?srsltid=AfmBOooDpdRV3TYyUPoC-n1kS8xDhkJ5H6fCNGBAau2iSObrxZQN6vY_

Photo shown may vary from your selections.

CAN Interface Device, 1- or 2-Port, High-Speed/FD

USB-8502

FROM € 1.133,00

Price may vary with addition of accessories and services.

Item details

Bus Connector: **USB 2.0**
CAN Physical Layer: **Flexible Data-Rate, High-Speed**

Select options

Number of CAN Ports

2

Supports External Timing And Triggering

Yes