Czech Technical University in Prague

Faculty of Mechanical Engineering

Department of Automotive, Combustion Engine and Railway Engineering

Master’s Thesis Title

HIL simulation of driving cycles and its validation

by

Bc. Harshaan Singh

Thesis supervisors:

doc. Ing. Jiří Novák, Ph.D., Czech Technical University, Prague
doc. Ing. Tomáš Haubert, Ph.D., Porsche Engineering Services, s.r.o., Prague

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Singh Harshaan**	Personal ID number:	**456141**
Faculty / Institute:	**Faculty of Mechanical Engineering**		
Department / Institute:	**Department of Automotive, Combustion Engine and Railway Engineering**		
Study program:	**Master of Automotive Engineering**		
Branch of study:	**Advanced Powertrains**		

II. Master's thesis details

Master's thesis title in English:

HiL simulation of driving cycles and its validation

Master's thesis title in Czech:

HiL simulace jízdních cyklů a její validace

Guidelines:

- learn about OBD diagnostics and HiL simulation method with focus on KOVOMO HiL system,
- implement simulation of standard driving cycle (e.g. WLTP) on HiL simulator focused on fuel consumption and (optionally) engine CO2/pollutant emissions,
- use an EXAM tool for test management/automation
- validate simulation results using oficially published data

Bibliography / sources:

Supke, P.: Diagnostic Communication with Road-Vehicles and Non-Road Mobile Machinery, SAE International 2019, ISBN 978-0-7680-9367-4
HiL simulator manuals, dSpace GmbH
ISO 15031, ISO_15765-3 standards, International Standardization Organization

Name and workplace of master's thesis supervisor:

doc. Ing. Jiří Novák, Ph.D., K 13138 - katedra měření

Name and workplace of second master's thesis supervisor or consultant:

Ing. Tomáš Haubert, Ph.D., Porsche Engineering Services, s.r.o.

Date of master's thesis assignment: **18.03.2020** Deadline for master's thesis submission: **14.08.2020**

Assignment valid until: ______________

doc. Ing. Jiří Novák, Ph.D.	doc. Ing. Oldřich Vítek, Ph.D.	prof. Ing. Michael Valášek, DrSc.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt   Student's signature

I dedicate this thesis to my nephew (Kuwarveer Singh) and my two nieces
(Mehr Kaur and Sohila Kaur)

Acknowledgement

Firstly, I would like to extend my gratitude to both my supervisors Dr. Jiri Novak at Czech Technical University, Prague and Dr. Tomas Haubert at Porsche Engineering Services for giving this opportunity to pursue my Master's Thesis on this exciting problem space.

I am extremely thankful to all my team members Mr. Martin Florian, Mr. Josef Nemet, Mr. Tomas Kamenca and Mr. Daniel Soares, of Electronics Integration Department for providing a good working atmosphere and assistance in day to day activities. I am fortunate to have spent my time at Porsche Engineering and gaining better understanding about Vehicle Diagnostics and Hardware-in-the-loop simulations.

Last but not the least, I would like to thank my family and friends for their immense support and love.

Abstract

To establish early niche in the market, vehicles being produced these days are growing exponentially in terms of complexity (hardware and especially in software). Keeping pace with this advancement, verifying and validating design becomes crucial steps. Hardware-in-the-loop (HIL) tests is one of the well adopted simulation test in the industry to overcome this challenge. HIL allows to test functionality and behavior of any vehicle component (any actuators or sensors) as though it is on the real vehicle, simulating all driving conditions, and identify all faults within any unit. In simple words, HIL replaces the need of assembled final product and hence comprehensive testing can be performed at early stage, giving the engineers and designer a head start.

Hardware-in-the-loop (HIL) simulation is used for all aspects of product development, including safety-relevant functions, simulating behavior of vehicle performance, etc. Nowadays, it is a standard component in the vehicle development process which provides various methods for testing of electronic control unit (ECU) software. All the vehicles physical parameters like temperature, air flow, vehicle speed, engine rpm, etc., are continuously monitored by electronic sensors and communicated, over the internal vehicle communications protocol, to the Main Control Unit for further processing.

This study present the selection of parameters used for calculation of the fuel consumption and prediction of CO_2 emissions on a simple driving cycle. These measurements are retrieved from Engine Control Module and OBD-II diagnostic protocol in case of HIL and real vehicle respectively. Comparing the driving cycle HIL data with data the real vehicle measurements, HIL is validated which help to understand the effects of various factors in the estimation of fuel consumption and CO_2 emissions. Further, using the results from this validation we can get clear depiction on how HIL will behave on WLTP cycle.

Keywords: Hardware-In-Loop, DSpace, Vehicle Diagnostics, Communication Protocols, EXAM, DiagRA, INCA, PIDs, On-Board Diagnostics (OBD-II), Diagnostic Trouble Code (DTC), CAN, FlexRay, LIN, Unified Diagnostic Service (UDS), WLTP, Volumetric efficiency, Short term fuel trim, Fuel Consumption, CO_2 emissions.

Table of Contents

Practical Part

List of Figures

List of Tables

Chapter 1
Introduction

This report introduces the master's thesis 'HIL simulation of driving cycles and its validation'. The work has been conducted at the Porsche Engineering Services, Prague, Czech Republic. This chapter presents the background, objectives, tasks and structure of the thesis.

1.1 Background

As per a report published by Harvard Kennedy School, it is estimated that worldwide the number of passenger cars will reach up to 1.5billion by 2025 compared to 750million in 2010 [1]. Following these growing numbers, in the past decade automotive industry has seen tremendous growth in the technological advancement in electrical vehicles and systems. One of the most important factor to continue this growth momentum is the parallel advancement in development and testing techniques.
There are few challenges when it comes to perform reliable and comprehensive testing. First and the foremost is the very high testing cost which then for most companies impacts their timelines/delivery schedules and subsequently the time to market. Another important challenge is the ability to achieve testing results at acceptable level of confidence in safety, quality and reliability.

Lately, hardware-in-the-loop (HIL) testing method has gained recognition and has become a principal part of control validation in the automotive product development cycle. According to ISO 26262 standard which is mandatory for passenger car development worldwide. It explicitly names HIL as a suitable test environment for software unit tests and integration tests, for the verification of safety requirements at component level and names it as a appropriate method for testing single ECUs/components and for testing ECU networks up to an entire virtual vehicle.

HIL simulation is rapidly progressing in an automotive industry from a control prototyping tool to a system modeling, simulation and synthesis methods which are combining many benefits of both physical and virtual prototyping. Vehicle is a very complex ecosystem consisting of multiple sub-processes/modules which are responsible for its smooth operations. Each module is monitored and controlled with the usage of sophisticated sensors which inform and collaborate with the Main Control Unit (MCU). The micro-controllers (supporting the sensors) communicate with the MCU and with each other using typical bus-based communications standards such as CANBus (Controller Area Network), Flexray, Local Interconnect network (LIN), etc.

To study the ECU efficient functionality, the HIL validation plays a very important role, so this thesis primarily focuses is on validating HIL using collecting simulated results from Engine Control Unit and validating them with the measurements derived from the vehicle (Model-Porsche Panamera). HIL being a very complex system architecture platform, it's validation in this study is mainly focused on the behavior of it in terms of fuel consumption calculation, as this gives us clear comparison about the engine efficiency. Another dimension of this study is comprehensive dive into in-vehicle architecture and vehicle diagnostics further used in HIL system validation on WLTP cycle focusing on fuel consumption.

1.2. Objectives

The objective of this work can be described in following points

1. First, to develop strong theoretical background and sound understanding of the foundation concepts in vehicle electronics and Hardware-In-Loop systems. The successful completion of this objective is very crucial to achieve rest of the objectives in this thesis.

2. The second objective is to gain expertise and proficiency in test automation software (Extended Automation Method (EXAM)) and diagnostics tool (DiagRA) used for HIL setup, specifically around functionalities and capabilities of both the software and system respectively.

3. The third objective is to validate HIL by focusing on fuel consumption with three phase driving cycle. The validation includes extracting the relevant data from the sensors and actuators attached to the Electronic control unit (ECU) and comparing this data with the real vehicle measurements via On-Board Diagnostics (OBD II) Scantool.

4. Lastly, using the understanding of the HIL behavior and sensitivity over varying, vehicle speed, rapid acceleration/ deceleration, and reasons behind it's over-under estimation of the results. The fuel consumption values via HIL are obtained on WLTP driving cycle.

1.3. Tasks of the thesis

With the aim to effectively fulfil the objective listed in 1.2, the thesis is organized under two major goals and further sub-divided into small tasks for continuous monitoring of progress.

I. Understanding of In-Vehicle network, communication protocols and vehicle diagnostics.

- Learn about On-Board and Off-Board Diagnostics and HIL simulation method with focus on dSpace KoVoMo HIL system which is used for all V6 and V8 Engines Porsche Vehicle.
- Acquire knowledge about vehicle diagnostics software and diagnostic trouble codes (DTC) and Parameter Identifiers (PIDs).

II. Hardware-In-Loop (HIL) testing and its validation using real vehicle measurements.

- WLTP Implementation of standard driving cycle using manual and automation approach on HIL system focused on fuel consumption and estimation of CO_2 emissions.
- Understanding and using of Extended Automation Method (EXAM) automation tool for test management/automation.
- Validate HIL and perform its data analysis using results from real vehicle.

1.4 Structure of the Master Thesis

The thesis work is organized in two parts: Theoretical and Practical Part.

The theoretical part covers the basics of Communication Network (Chapter 2), Vehicle Diagnostics (Chapter 3), and Porsche Hardware-In-Loop (HIL) setup (Chapter 4).

The Practical part starts with giving insights of DiagRA D–Diagnostic Software tool and EXAM (EXtended Automation Method) is a test management system software (Chapter 5). Chapter 6 describes Worldwide Harmonized Light Vehicles Test Procedure (WLTP), its implementation (manual and automated) and HIL Virtual Driver Behavior.

Chapter 7 generated the mathematical model with assumption in order to calculate fuel consumptions and CO_2 emissions. Chapter 8 discusses about HIL Validation using real vehicle measurements. Based on the data analysis of the consumption and CO_2 estimations, various cause of the effects is discussed later in this chapter. Further, the results obtained via WLTP cycle on HIL is compared using official published data.

Finally, the last chapter 9 summarizes the final conclusion of the thesis.

Theoretical Part

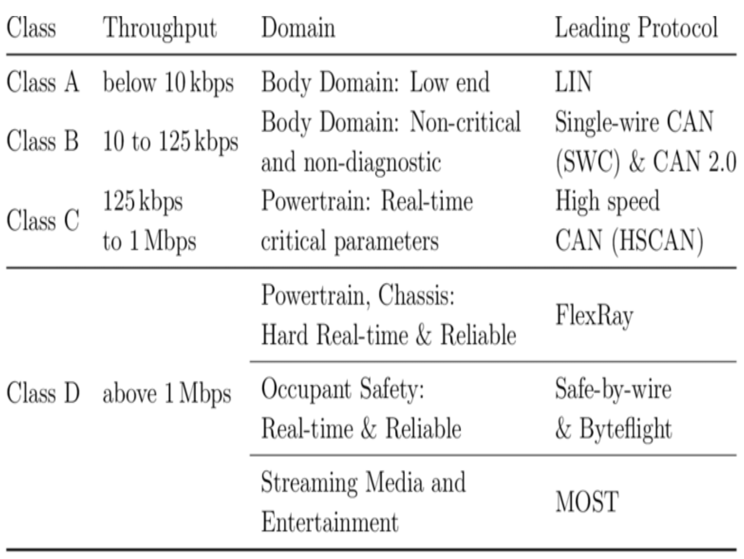
Chapter 2
Communication Networks

In the chapter in-vehicle network which is a special internal communication network that interconnects components inside the vehicle are discussed. Further, the hardware aspects of in-vehicle networking and its main standards for e.g. CAN, LIN, FlexRay and Ethernet are explained as these communication protocol is widely used by Porsche Engineering Services Hardware-in-Loop system.

2.1 In-Vehicle Networks

Electronic safety-critical control function in vehicles was first used in 1981. General Motors implemented micro-computer based engine control for their petrol powered vehicles which greatly improved the efficiency and performance [2]. With the introduction of laws regulating emission control, the use of electronic engine control (ECU) was required to meet the legal requirements as well as to maintain acceptable efficiency and performance. The ease of implementation along with the cost/efficiency benefits motivated manufacturers to adopt electronic control for engine management and this later spread to other domains.
Currently, in modern vehicles around 30–50 ECUs across all segments are to be found. These ECUs consist of automotive grade micro-controllers and/or general purpose processors which execute software implementations for control and comfort applications. The number of ECUs in vehicles has been rising at the rate of approximately 1.45 times a year, while the application software has been growing at a rate of 4.5 MB per year.

Depending on the domain the ECU is intended for, suppliers also provide customized architectures that are best suited for functionality in that specific domain. For example, a body domain controller might be working on different network protocols and offer little or no hardware acceleration support, while a telematics controller would integrate high speed interconnect and dedicated accelerator blocks for video processing or radar interfaces. This "right-sizing" enables manufactures to control the cost (development and parts) as well as standardize the software framework for each domain. The Society for Automotive Engineers (SAE) classifies in-vehicle networks based on throughput and domain of operation [3] as shown in Table 1.



Class	Throughput	Domain	Leading Protocol
Class A	below 10 kbps	Body Domain: Low end	LIN
Class B	10 to 125 kbps	Body Domain: Non-critical and non-diagnostic	Single-wire CAN (SWC) & CAN 2.0
Class C	125 kbps to 1 Mbps	Powertrain: Real-time critical parameters	High speed CAN (HSCAN)
Class D	above 1 Mbps	Powertrain, Chassis: Hard Real-time & Reliable	FlexRay
		Occupant Safety: Real-time & Reliable	Safe-by-wire & Byteflight
		Streaming Media and Entertainment	MOST

Table 1: SAE in-vehicle network classification.

2.2 Communication Protocols

Every modern vehicle use different network protocols in different domains, the choice of which is determined by factors such as the functional requirements of the domain, criticality, cost, etc. Among the many protocols, Local Interconnect Networks (LIN), Controller Area Networks (CAN), FlexRay, and Media Oriented Systems Transport (MOST) are the most widely used protocols by the different manufacturers today. Special networks like safe-by-wire are used for passenger safety systems like airbags and other active protection systems. A simplified scheme of typical in-vehicle network architecture in a modern vehicle is as shown in Figure 1.

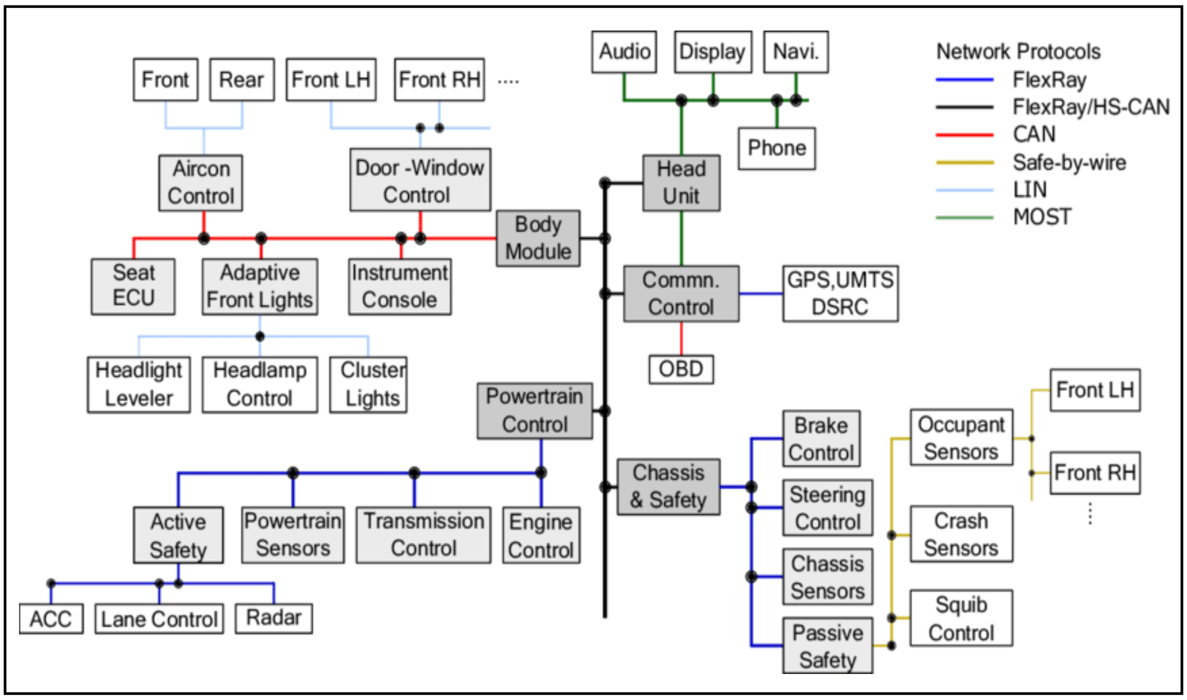


Fig 1: In-vehicle network architecture [2].

A large variety of in-vehicle networks evolved primarily due to cost and performance requirements. CAN is very expensive and complicated for simple functions like power windows or boot release. Simpler protocols like the Local Interconnect Network (LIN) is adoption due to its non-critical functionality at lower cost per module and power consumption. While, CAN is too slow for high bandwidth applications like multimedia in higher end vehicles resulting in the development of high bandwidth protocols like Media Oriented Systems Transport (MOST) for such applications. Time-triggered CAN (TTCAN) is an evolution of standard CAN, which addresses the lack of its functionality by introducing a time-triggered mechanism above the CAN framework. The FlexRay protocol, developed by the FlexRay consortium, offers a combination of time-triggered and event-triggered communication for in-vehicle applications to enhance reliability with higher bandwidth and is mostly used in Porsches.

2.2.1 Controller Area Network (CAN)

CAN (Controller Area Network) bus is one of the most popular protocols in the automotive industry, which enables different components of vehicles to communicate with each other. It was established by Robert Bosch in 1983 and officially released in 1986. It handles a maximum signaling rate of 1 megabit per second (bps). CAN is an International Standardization Organization (ISO-11898: 2003) defined serial communications bus, originally developed for the automotive industry. It is a two-wire (twisted pair) communications bus and has a high immunity to electrical interference and can self-diagnose and repair data errors.
The ISO-11898 [4] standard defines CAN by using the Open Systems Interconnection (OSI) model which is defined in terms of layers. Figure 2 shows, the two lowest layers of the seven layer OSI model: the data-link and physical layer and ISO 15765-2 [5] specifies Transport and Network layer services.

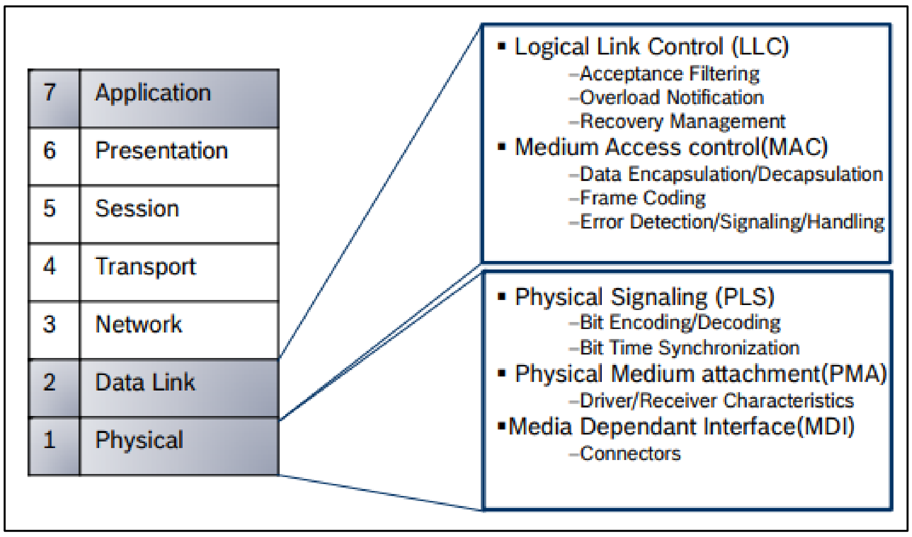

7 Application
6 Presentation
5 Session
4 Transport
3 Network
2 Data Link
1 Physical
Logical Link Control (LLC)
–Acceptance Filtering
–Overload Notification
–Recovery Management
Medium Access control(MAC)
–Data Encapsulation/Decapsulation
–Frame Coding
–Error Detection/Signaling/Handling
Physical Signaling (PLS)
–Bit Encoding/Decoding
–Bit Time Synchronization
Physical Medium attachment(PMA)
–Driver/Receiver Characteristics
Media Dependant Interface(MDI)
–Connectors

Fig 2: CAN Bus OSI Model.

The protocol used for CAN is the carrier-sense, multiple-access with collision detection (CSMA/CD). The arbitration is based on the message priority and is implemented on bit level (bit-wise arbitration). The node with the highest priority identifier which is accomplished by longest dominant bit levels in the identifier, prioritized as the bus access.

CAN BUS FRAME

Four different CAN messages exist in the CAN protocol [6], explained as followed:

Data frame: The CAN data frame also works with two different protocols. The first one is called "base format" and has an identifier of 11 bits. The second one is the "extended format" and the identifier has 29 bits. The standard says that a CAN controller must accept at least basic frames but can or cannot accept extended frames.
Remote frame: It works the same as the previous one but there is a difference. It is possible that a node requires some data from another one. Then, a remote frame is requested to the second one in order to get the information. Basically, the difference between data frames and remote frames is that the last ones do not have data field.

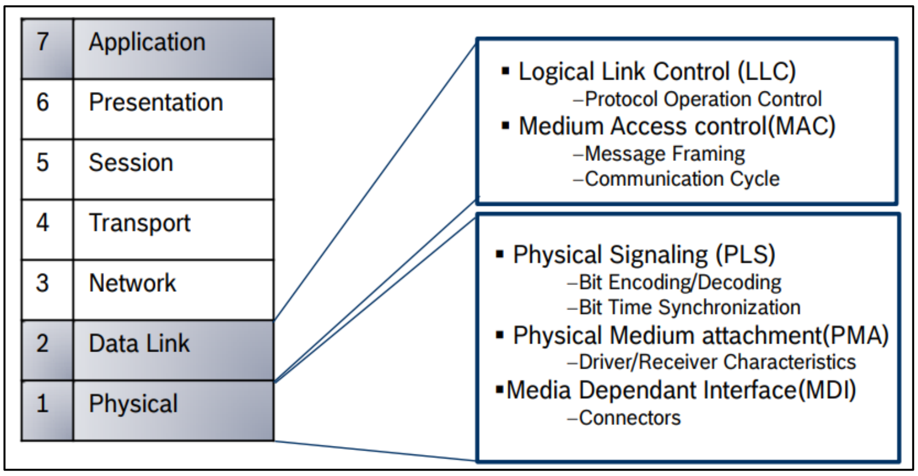
Error frame: This is a special frame that is transmitted when a node detects a wrong message. Then, the rest of nodes also transmit an error frame. There is an error counter that avoids the blockade of the bus with continuous errors.
Overload frame: It is similar to error frame and is transmitted by a node when it is very busy. Then the bus starts providing extra delays between the CAN messages. For further explanation about CAN frame, please refer to the appendix.

2.2.2 FlexRay

The FlexRay communications bus is a deterministic, fault-tolerant and high-speed bus system developed in correspondence with automobile manufacturers and leading suppliers. FlexRay delivers the error tolerance and time-determinism performance requirements for x-by-wire applications where x can be drive-by-wire, steer-by-wire, brake-by-wire, etc.

One of the things that differentiates FlexRay, CAN and LIN from more traditional networks such as Ethernet, is its topology, or network layout. FlexRay supports multi-drop passive for simple connections as well as active star connections for more complex networks. In contrast, when FlexRay is configured to talk on a bus, it uses something called a time division multiple access (TDMA) scheme to guarantee determinism. Its node is synchronized with the same clock and each node waits until it is the turn to write to the bus. As the timing in a TDMA scheme is consistent, it can guarantee determinism or the consistency of data delivery to nodes in the network. FlexRay devices cannot automatically detect the network or addresses on the network, so it is essential to have that information programed in at manufacturing time.

The ISO-17458 [7] standard defines FlexRay by using the Open Systems Interconnection (OSI) model which is defined in terms of layers. Figure 3 shows, the two lowest layers of the seven layer OSI model: the data-link and physical layer and ISO 10681-2 [8] specifies Transport and Network layer services.


7 Application
6 Presentation
5 Session
4 Transport
3 Network
2 Data Link
1 Physical
▪ Logical Link Control (LLC)
–Protocol Operation Control
▪ Medium Access control(MAC)
–Message Framing
–Communication Cycle
▪ Physical Signaling (PLS)
–Bit Encoding/Decoding
–Bit Time Synchronization
▪ Physical Medium attachment(PMA)
–Driver/Receiver Characteristics
▪Media Dependant Interface(MDI)
–Connectors
Fig. 3: FlexRay OSI Model

FLEXRAY FRAME

The FlexRay frame consists of the three segments, the header segment, the payload segment and the trailer segment.

Header Segment: The FlexRay header segment consists of five bytes (40 bits). These bytes contain a reserved bit, the payload preamble indicator, null frame indicator, sync frame indicator, frame ID, startup frame indicator, payload length, header CRC and the count for cycles.

Payload Segment: The FlexRay payload segment comprises of 0 to 254 bytes data. The bytes are identical numerically, starting at Data 0 for the first byte after the header segment increasing by one with each subsequent byte.
For frames communicated in the static segment the first 0 to 12 bytes of the payload segment may optionally be used as a network management vector. The payload preamble indicator in the frame header shows whether the payload segment contains the network management vector. The length of the network management vector can be configured from 0 to 12 bytes.
For the frames transmitted in the dynamic segment the first two bytes of the payload segment can be used as a message ID field, allowing the receiving nodes to filter data based on the contents of this field. The payload preamble indicator in the frame header indicates whether the payload segment contains message ID.

Trailer Segment: The FlexRay trailer segment comprises of a single 24 bit field. This has CRC calculations values which have been calculated by the host for the fields in the header and the payload segments for the field.
For further explanation about FlexRay frame, please refer to the appendix.

2.2.3 Local Interconnect Network (LIN)

The LIN consortium comprises many vehicle manufacturers like Audi, Volvo, and BMW. LIN is a cheap slow serial bus used for distributed body control electronic systems in vehicle. It enables effective communication for sensors and actuators where bandwidth, speed and versatility are not required (i.e inside mechatronic based subsystems generally made of an ECU and its set of sensors and actuators). LIN is usually used as a sub bus for CAN and Flexray.

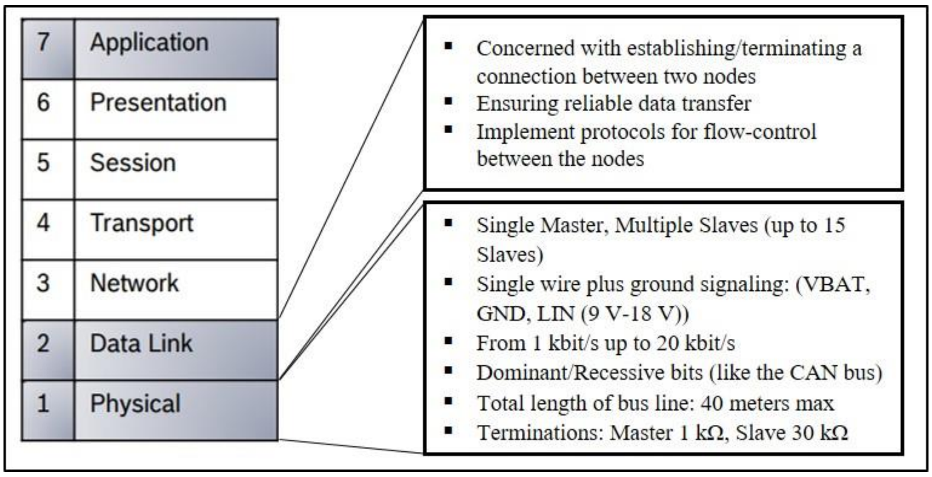

7 Application
6 Presentation
5 Session
4 Transport
3 Network
2 Data Link
1 Physical
Concerned with establishing/terminating a connection between two nodes
Ensuring reliable data transfer
Implement protocols for flow-control between the nodes
Single Master, Multiple Slaves (up to 15 Slaves)
Single wire plus ground signaling: (VBAT, GND, LIN (9 V-18 V))
From 1 kbit/s up to 20 kbit/s
Dominant/Recessive bits (like the CAN bus)
Total length of bus line: 40 meters max
Terminations: Master 1 kΩ, Slave 30 kΩ

Fig. 4: LIN OSI Model

The ISO-17987 [9] standard defines LIN by using the Open Systems Interconnection (OSI) model which is defined in terms of layers. Figure 4 shows, the two lowest layers of the seven layer OSI model: the data-link and physical layer ISO-17987-3 and ISO-17987-4 and ISO 15765-2 specifies Transport and Network layer services.

LIN FRAME

The LIN bus is a polled bus with a single master device and one or more slave devices [10]. The master device has both, a master task and a slave task. Each slave device contains only a slave task. Communication over the LIN bus is controlled totally by the master task which is in the master device. Frame is divided into a header and a response which is the basic unit of transfer on the LIN bus. The header is always transmitted by the master node and it consists of three separate fields: the break, synchronization (sync), and identifier (ID). The response is transmitted by a slave task which resides in either the master node or a slave node. It contains a data payload and a checksum.

Normally, the master task analyze each slave task in a loop by transmitting a header, which consists of a break-sync-ID sequence. Before starting the LIN, each slave task is designed to either publish data to the bus or subscribe to data in response to each received header ID. When the header is received, each slave task verifies ID similarity and then checks the ID to decide whether it needs to publish or subscribe. If the slave task wants to publish a response, it transmits 1-8 data bytes to the bus after that by a checksum byte. If the slave task wants to subscribe, it reads the data payload and checksum byte from the bus and takes appropriate internal action.

For standard slave-to-master communication, the master transmits the identifier to the network, and just one slave responds with a data payload.

Master-to-slave communication is done by a separate slave task in the master node. This task receives all published data to the bus and responds as if it were an autonomous slave node. To transmit data bytes, the master should first update its internal slave task’s response with the data values it wants to communicate. The master then issues the suitable frame header, and the internal slave task then sends its data payload to the bus. Further explanation is in the appendix.

2.3 Future Communication Protocol

The automotive network architecture is currently facing the boundaries of established technology. The gradually increasing need for bandwidth and the diversification of performance, costs and dependability requirements lead to a modification of the networks used throughout the vehicle. Traditional protocols such as CAN, Flexray and LIN do not meet the bandwidth and scalability requirements, for example the Advanced Driver Assistance Systems (ADAS).
Around 2500 signals in today's luxury vehicles (i.e. elementary information such as the speed of the vehicle) are exchanged by up to 70 ECUs [11]. Until the start of the 90s, the data was exchanged through point-to-point links between ECUs. However this strategy, which required an amount of communication channels of the order of n^2 where n is the number of ECUs (i.e., if each node is interconnected with all the others, the number of links grows in the square of n), was unable to handle with the increasing use of ECUs due to the problems of weight, cost, complexity and reliability induced by the wires and the connectors. CAN FD and Ethernet are in nearly all vehicles currently in mass production in VW. For this study, CAN FD communication protocol is used to retrieve measurement data from DiagRA Software while FlexRay is used to flash software on the ECU.

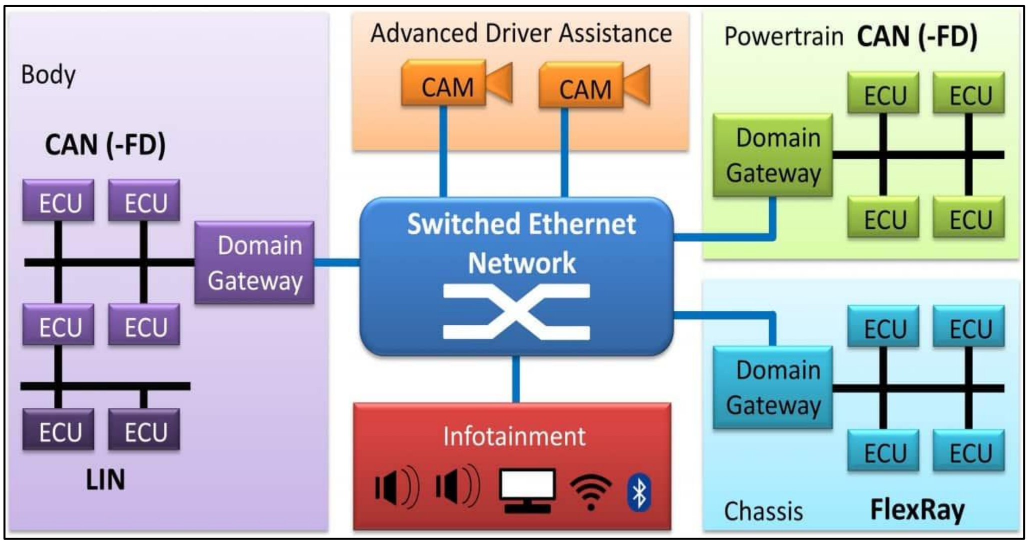


Fig. 5: Future automotive backbone network

Ethernet and CAN-FD is the emerging technology in the automotive domain. It is capable to address bandwidth demands of tomorrow's advanced driver assistance systems (for example, HD video, LIDAR) and it will also provide greater interoperability with consumer multimedia products such as smartphones and tablets. In the following article, two of the new automotive networking protocols, CAN-FD and Ethernet are discussed.

2.3.1 Controller Area Network Flexible Data-Rate (CAN-FD)

CAN FD was developed in 2011 by Robert Bosch GmbH, in Germany as an addition to the original CAN protocol. Working closely with the prominent carmakers and other CAN experts and answering to the need of the more powerful CAN protocol, Bosch came up with CAN-FD. The "FD" in CAN FD means "flexible data-rate," which is the big development, allowing increased performance and higher bandwidth communication. This new and improved extension to the Standard CAN protocol allows for data transfers of 8 MB/s, even with cable lengths more than 40 meters. It can transfer up to 64 bytes of data in a single message.

CAN-FD FRAME

The CAN-FD frame format is shown in Figure 6. Similar to CAN as discussed in the previous section, CAN-FD **dominant bit** is a logical 0 and a **recessive bit** is a logical 1. As shown in the figure, a CAN-FD frame is consist of two phases: arbitration phase and data phase [12].

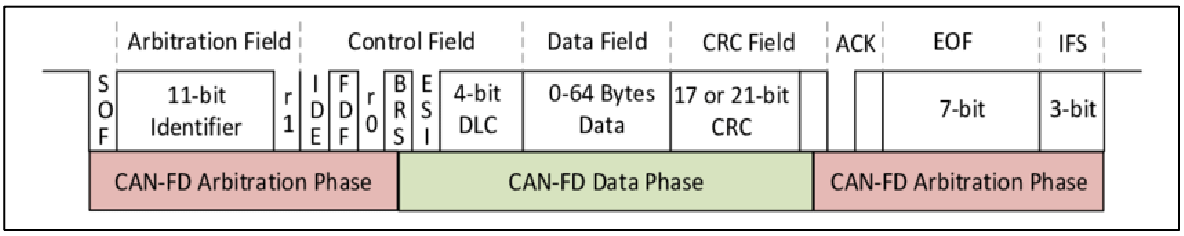


Fig. 6: CAN-FD Frame format

Arbitration Phase: The arbitration phase in the CAN-FD frame consist of: SOF (Start of Frame), arbitration, part of the control field, ACK (Acknowledgment), EOF (End OF Frame), and IFS (Inter-Frame Space). The 11-bit (or 29-bit in case of extended format) identifier represents the priority of the frame: the lower the value of the identifier, the higher the priority. The arbitration for transmission happens as follows:
During the idle state of the bus, all the nodes with some ready frames send the 11-bit identifier after the SOF bit. During the transmission of the identifier bits, if a node transmits a recessive bit but finds a dominant bit on the bus, it stops transmission due to the presence of a higher priority node contesting for transmission. In the end, the node with the highest priority message wins the arbitration and continues the transmission.

Data Phase: The BRS (Bit-Rate Switch) bit is one of the add-ons to the CAN-FD frame format. It is used to decide whether the bit-rate in the data phase is the same as that of the arbitration phase (BRS = 0) or it switches to the increased bit rate (BRS = 1). Since the focus is on CAN-FD, the BRS bit in the frames to be recessive (i.e., BRS = 1) is considered. At the increased rate of data transmission, each bit transmission occurs with a duration denoted by t_d.
For example, if the data rate is chosen as 2 Mbps, $t_d = 0.5\mu s$. The 4-bit DLC (data-length code) field stipulates the payload size (in bytes) of the data field. CAN-FD offers 16 separate payload sizes: 0 through 8, 12, 16, 20, 24, 32, 48 and 64 bytes.
The data field is followed by the Cyclic Redundancy Check (CRC) field, which has 17 bits for payloads up to 16 bytes, and 21 bits otherwise. The CRC delimiter bit (recessive) is transmitted next. After this, the bit rate is reversed to that of the arbitration phase.

2.3.2 Automotive Ethernet

Ethernet is the evolving technology in the automotive industry. Due to its greater bandwidth and flexibility and the promise of sharing cost of ownership with other industrial segments. Ethernet is perfect to address the high demands of new functions in infotainment and advanced driver assistance systems or to decrease ECU flashing speed and updating cost. With the first generation of vehicles using Ethernet as an added communication medium, it is also considered as a very powerful backbone advancement in the future technology which is also capable of carrying traffic originating in CAN (-FD) or other bus subsystems as shown in figure 7.

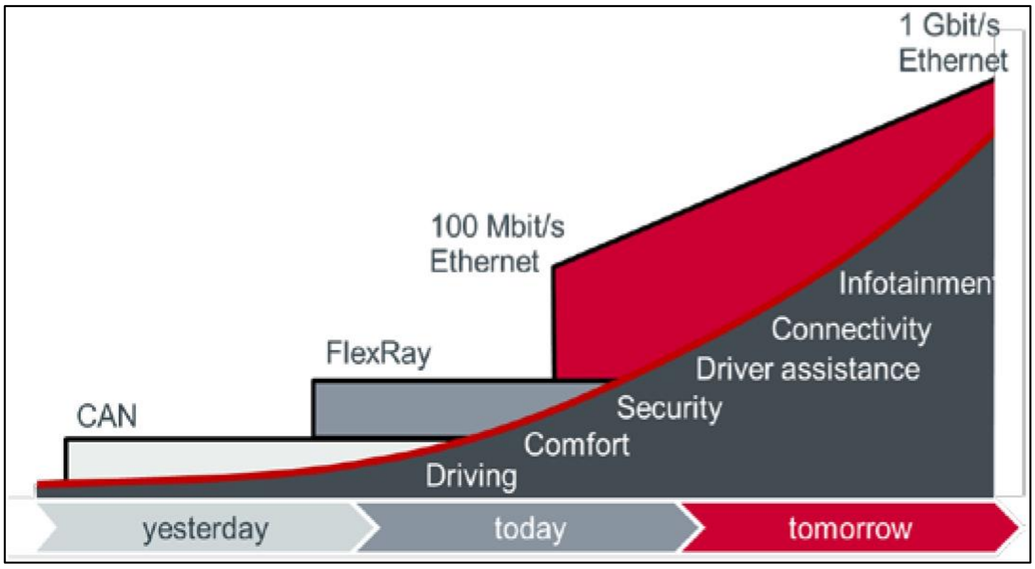


Fig. 7: The fast-growing demand for bandwidth

ETHERNET FRAME

An Ethernet frame is a piece of data along with the information that is required to transport and deliver specific piece of data. In networking reference models, such as; OSI Seven Layers model and TCP/IP, the Ethernet frame is defined in the Data link layer same as CAN, LIN and FlexRay.

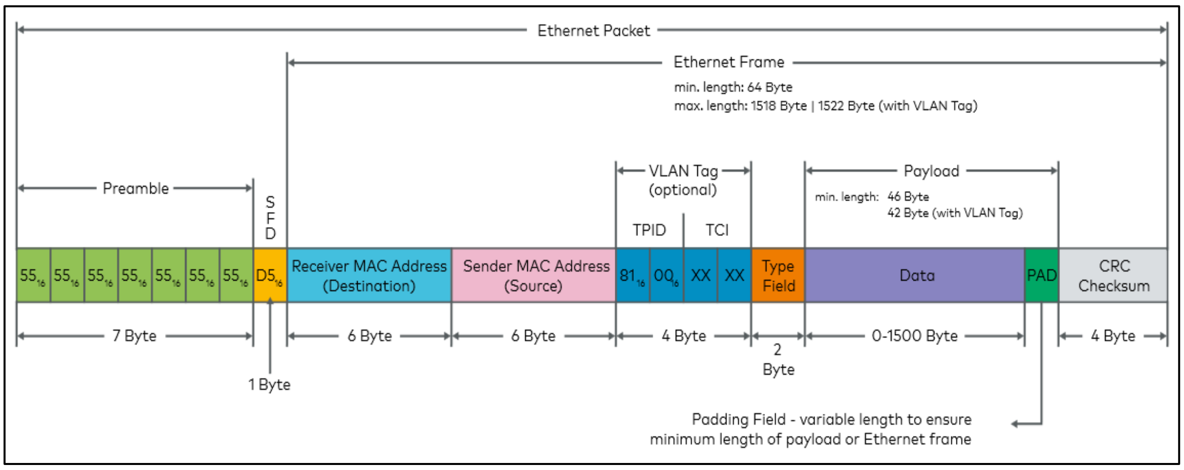


Fig. 8: Ethernet Frame format [13]

The basic frame consists of seven elements divided in three main areas as shown in figure 8:

Header

Preamble / SFD - This element in header is added by the layer 1 part of the protocol stack. It enables the receiver to synchronize and know that a data frame is about to be sent.

- Preamble (PRE) - This is seven bytes long and it consists of a pattern of alternating ones and zeros, and this informs the receiving stations that a frame is starting as well as enabling synchronization.
- Start of Frame Delimiter (SFD) - This consists of one byte and contains an alternating pattern of ones and zeros but ending in two ones.

Destination Address (DA) - This field consist of the address of station for which the data is intended for. The left most bit shows whether the destination is an individual address or a group address. An individual address is denoted by a zero, while a one is for a group address. The next bit in the DA is to understand whether the address is globally administered or local. If the address is globally administered then the bit is zero valued, and a one is when it's locally administered. There are then 46 remaining bits. These are used for the destination address itself.

Source Address (SA) - The source address comprises of six bytes and it's used to recognize the sending station. Being an individual address, the left most bit is always valued as a zero.

Length / Type - This field length consist of two bytes. It offers MAC information and specifies the number of client data types that are contained in the data field of the frame. If the frame is assembled using an optional format (IEEE 802.3 only) in that case it may also indicate the frame ID type.

VLAN tag - It contains a protocol identifier (TPID) and control information (TCI). While the TPID consist of original type field value, the TCI comprises of a Priority (PCP), a Drop Eligible or Canonical Form Indicator (DEI or CFI) and an Identifier (VID). VID and PCP are mainly used in the automotive industry. The Identifier separates the respective virtual network for the different application areas. The Priority allows optimization of run-times through switches so that important information is sent preferentially.

Payload

Data - This block consist of the payload data and it can be up to 1500 bytes long. Padding data is added to increase its length up to the required minimum of 46 bytes, in case if the length of the field is less than 46 bytes.

Trailer

Frame Check Sequence (FCS) - FCS is four bytes long. It consist of a 32 bit Cyclic Redundancy Check which is generated over the Destination Address, Source Address, Length / Type and Data fields.

Chapter 3
Vehicle Diagnostics

In this chapter, on-board and off-board diagnostics signal protocols, Parameter IDs and understanding trouble codes (DTCs) are explained. Further, basics of software Integrated Calibration and Application Tool (INCA) along with very commonly used terms in the diagnostics are described.

3.1 Introduction

Diagnostic determines, verifies and classifies which is focused to get an overall picture in finding the root cause of a problem in a vehicle. The detection, improvement and communication strategies applied to irregular operation of systems is examined by Electrical and electronic devices. Therefore, the purpose of Diagnostic is to identify this root cause of irregularities in its operation so a restoration can be performed. Diagnostic requirements for OEM and supplier are defined by a common database which contains the functional diagnostic requirements, its implementation, development, specific data concerning to it and also its features. Every industry have a straight connection with product engineering, manufacturing, aftersales and suppliers. Applications of diagnostic can be classified for the following fields as OEM [14]:

Development – In this process, correct functionality of the vehicle's components must be authenticated. Then subsystem of the diagnostic takes part at reading out ECU's internal information and data of sensor and actuator's values.
Production – The assembly plant uses this system for transferring calibrated/authenticated data and software updates to the non-volatile memory of the ECUs, including EOL programming and tests.
Aftersales – In the operating vehicle, error detection is mainly done via diagnostics. Detected errors are stored to a persistent fault memory, and trouble codes are read out at the service station in order to make troubleshooting possible. The diagnostic systems include both on-board diagnostics and off-board diagnostics discussed as follow.

3.2 On-Board diagnostics (OBD)

OBD is the computer system built into vehicles that monitors the performance of the engine components. It consists of several ECUs that uses various sensors to collect data and evaluate the performance of the vehicle as shown in figure 9. The OBD system will detect problems with the vehicles performance or functions before the problems become noticeable to the driver. These services can perform tests that can control actuators and read sensor values in the vehicle. This diagnostics can also continuously monitor sensor values and the state of the vehicle, whenever the fault occurs in the vehicle trouble codes are generated, called DTCs.

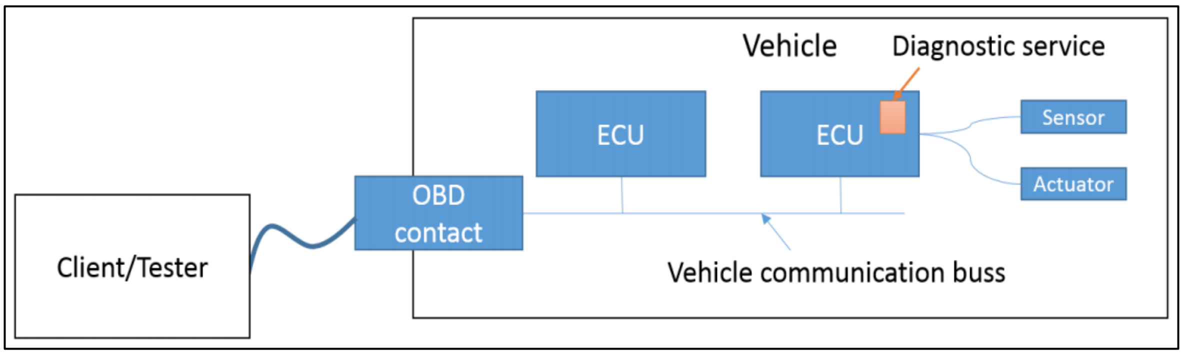


Fig. 9: On- board vehicle diagnostics. Diagnostic tester/client connected to a vehicle to run diagnostic services in an ECU

OBD-I mentions about the first generation of diagnostics which was developed during the 1980s, at that time, due to a lack of standardization, every vehicle manufacturer used different connectors and communication protocols. OBD-II also written as OBD2, is the successor to OBD-I and was developed in the early 1990s by the American organization Society of Automotive Engineers (SAE) which ordered all compliant vehicles to use a standardized connector and one of several standardized communication protocols [15]. European On-Board Diagnostics (EOBD) is the European version of vehicle diagnostics and is technically comparable to OBD II but was not implemented until 2001 for petrol vehicles and 2004 for diesel vehicles [16].
The standard requires that vehicles should have a 16-pin OBD II port. Sensor data and diagnostic information from the electronic control unit (ECU) of a vehicle is measured or extracted from this port. SAE J1962 [17] defines the pinout of the connector as shown in Figure 10. There are two types of connector: Type A and Type B connector, the nominal supply voltage at the contact 16 and the supported current supply in case of type A should be 12 V DC and 4,0 A while in type B it should be 24 V DC and 2,0 A respectively.

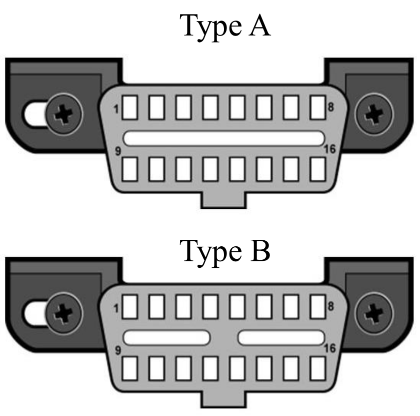


Fig. 10: Vehicle connector and contacts allocation

CONTACT	GENERAL ALLOCATION
1	Discretionary
2	Bus positive line SAE J1850
3	Discretionary
4	Chassis Ground
5	Signal Ground
6	CAN_H Line of ISO 15765-4
7	K Line of ISO 9141-2 and ISO 14230-4
8	Discretionary
9	Discretionary
10	Bus negative line of SAE J1850
11	Discretionary
12	Discretionary
13	Discretionary
14	CAN_L line of ISO 15765-4
15	L line of ISO 9141-2 and ISO 14230-4
16	Permanent positive voltage

3.2.1 OBD-II Signal Protocols

The development of OBD II also lead to in the development of OBD II scanning tools, like OBD II readers, which can communicate to any vehicle via the 16-pin port. A scanning tool normally requests information from the ECU by sending a message comprising of a hexadecimal code connected with a specific parameter. These codes are defined by the SAE J1979 standard (explained further in the document). The message would then get interpreted according to one of five mainly used OBD II signaling protocols discussed as follows:

STANDARD	DESCRIPTION
SAE J1850 PWM (pulse-width modulation 41.6 kB/sec)	• Pin 2: Bus+ • Pin 10: Bus– • High voltage is +5 V • Message length is restricted to 12 bytes, including CRC (cyclic redundancy check)
SAE J1850 VPW (variable pulse-width-10.4/41.6 kB/sec)	• 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
ISO 9141-2 (Similar to Recommended std. RS-232)	• Pin 7: K-line • Pin 15: L-line (optional) • UART (universal asynchronous receiver-transmitter) signaling • K-line idles high, with a 510 ohm resistor to V_{batt} • The active/dominant state is driven low with an open-collector driver. • Message length is Max 260Bytes. Data field MAX 255.
ISO 14230 KWP2000 (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 (250 kBit/s or 500 kBit/s)	• Pin 6: CAN High • Pin 14: CAN Low

3.2.2 Parameter Identification Numbers (PIDs)

OBD II uses two types of codes to request ECU data, these are Diagnostic Trouble Codes (DTCs) and Parameter Identifiers (PIDs). DTCs (for more details refer to section 3.2.3) are used to diagnose malfunctions in various subsystems of the vehicle and PIDs (hexadecimal code) are used to measure real time parameters. Vehicle manufactures have power to define their own PIDs by this means making the on-board system more sophisticated.

These codes are defined by the SAE J1979 standard [Table 2 below].

DIAGNOSTIC SERVICE MODE OF OPERATION	**DESCRIPTION**
$01	Request Current Powertrain Diagnostic Data
$02	Request Powertrain Freeze Frame Data
$03	Request Emission-Related Diagnostic Trouble Codes
$04	Clear/Reset Emission-Related Diagnostic Information
$05	Request Oxygen Sensor Monitoring Test Results
$06	Request On-Board Monitoring Test Results for Specific Monitored Systems
$07	Request Emission-Related Diagnostic Trouble Codes Detected During Current or Last Completed Driving Cycle
$08	Request Control of On-Board System, Test or Component
$09	Request Vehicle Information
$0A	Request Emission-Related Diagnostic Trouble Codes with Permanent Status

Table 2: Purpose of each mode of operation. The dollar sign “$” in front of the numerical value highlights that this is an identifier. It’s important to know that the numerical values of the identifiers are in hexadecimal format.

The message is interpreted according to one of five OBD II signaling protocols. The ECU sends a hexadecimal code in response. Depending on the specific parameter being measured, the real measurement can be extracted by simply converting the returned hexadecimal value to decimal or by carrying out a calculation using a standard formula as defined in [18] [19] for that specific parameter.

For most modes (explained above) there are several PIDs defined that specifies the request in more detail. For example mode 01, PID 0D requests the current vehicle speed and mode 09 PID 02 requests the Vehicle Identification Number (VIN). Some modes do not require a PID, for example, mode 03 requests the stored trouble codes (DTCs) and mode 04 clears it from memory. Every PID has a defined response that is expected from the request. The responses are defined in SAE J1979 [18] [19] and describes in detail what the response should be, how many bytes the response contains and how the data is encoded in those bytes.

3.2.3 Diagnostic Trouble Codes (DTCs)

DTC stands for Diagnostic Trouble Code. It is used to identify faults in nodes. This is the foundation of Diagnostics. When fault is occurred in the vehicle, connected ECU captures it and stores it in memory as fault code. This is specific number for type of fault is called Diagnostic Trouble Code. This information can be retrieved either by tools at service station (e.g. OBD2 Scantool) or by in vehicle methodologies.

- **DTCs STRUCTURE**

DTCs have 4 bytes, 3 bytes to identify them and 1 byte to denote the current status of the DTC as shown in figure 11 below.

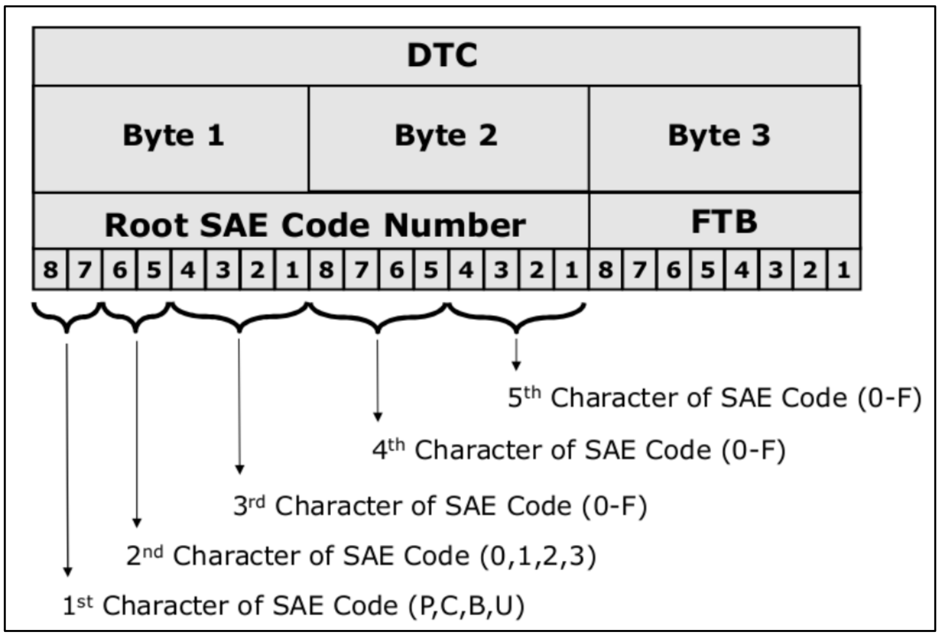

DTC
Byte 1
Byte 2
Byte 3
Root SAE Code Number
FTB
8 7 6 5 4 3 2 1 8 7 6 5 4 3 2 1 8 7 6 5 4 3 2 1
5th Character of SAE Code (0-F)
4th Character of SAE Code (0-F)
3rd Character of SAE Code (0-F)
2nd Character of SAE Code (0,1,2,3)
1st Character of SAE Code (P,C,B,U)

Fig. 11: DTCs

Byte - 1 and Byte - 2: To Identify the failed component - called as "ROOT DTC"
First two bits help identify the major system:
00 = P - Code for Powertrain
01 = C - Code for Chassis
10 = B - Code for Body
11 = U - Code for Network

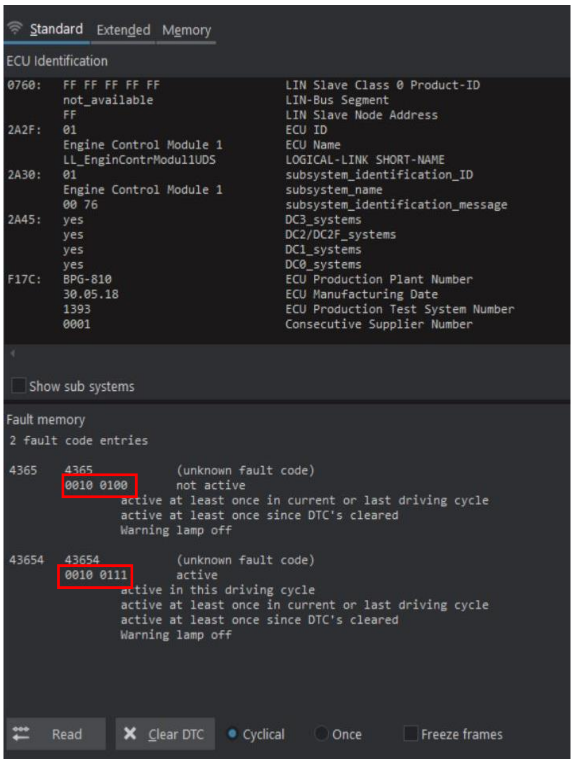
Byte - 3: Failure Type Byte ("FTB") – To identify failure mode of the ECU.
There are lot of FTBs. ISO-15031-6 has a list. Common Codes are 11 for short circuit to ground, 13 for open circuit.

Byte - 4: Status Information - Each DTC will have Status byte that provides the status information of DTC. Each bit in this bytes has a meaning and provides different information. This byte is widely used extract error information while performing the various testing scenarios at Porsche Engineering using DiagRA software figure 12.

There are 8 different states explained as follow:

Bit0: This Bit is “testFailed”. This bit gives the information about the fault (Error) is still active (injected) or not. If Fault is still Active/injected, then the value is 1 otherwise the value is 0.

Bit1: This Bit is “testFailedThisOperation Cycle”. This bit specifies whether the fault has occurred anytime during the current operation cycle. If Fault has occurred in the present operation cycle, then the value is 1 otherwise the value is 0.



```
0760:   FF FF FF FF FF                LIN Slave Class 0 Product-ID
        not_available                 LIN-Bus Segment
        FF                            LIN Slave Node Address
2A2F:   01                            ECU ID
        Engine Control Module 1       ECU Name
        LL_EnginContrModul1UDS        LOGICAL-LINK SHORT-NAME
2A30:   01                            subsystem_identification_ID
        Engine Control Module 1       subsystem_name
        00 76                         subsystem_identification_message
2A45:   yes                           DC3_systems
        yes                           DC2/DC2F_systems
        yes                           DC1_systems
        yes                           DC0_systems
F17C:   BPG-810                       ECU Production Plant Number
        30.05.18                      ECU Manufacturing Date
        1393                          ECU Production Test System Number
        0001                          Consecutive Supplier Number
```

```
2 fault code entries

4365    4365            (unknown fault code)
        0010 0100       not active
                  active at least once in current or last driving cycle
                  active at least once since DTC's cleared
                  Warning lamp off

43654   43654           (unknown fault code)
        0010 0111       active
                  active in this driving cycle
                  active at least once in current or last driving cycle
                  active at least once since DTC's cleared
                  Warning lamp off
```

Fig. 12: Byte 4 (red box)

Bit2: This Bit is “pendingDTC”. This bit informs whether the fault has occurred anytime during the current operation cycle. The only difference between “Bit1” and “Bit2” is that Bit1 is cleared at the end of current operation cycle (it does not bother whether the fault is still active or not) and “pendingDTC” is cleared only when in the succeeding operation cycle, monitor routine is run and the end result shows fault is absent (pass). So if Fault is still present in the current operation cycle, then the value is 1 otherwise if the Fault was active in previous operation cycle and is inactive in the present operation cycle, then the value is 0.

Bit3: This Bit is “confirmedDTC”. This bit informs that fault is constantly active for specific monitor routines and is matured enough in the existing operation cycle so that it can be said confirmed. If fault is active and matured, then the value is 1 otherwise it is 0.

Bit4: This Bit is “testNotCompletedSinceLastClear”. This bit notifies that monitor routine is not to be run in the existing operation cycle (once after Clearing the DTC is done). The reason being because particular pin is inactive in the operation cycle (e.g. parked or hibernate vehicle mode). If the monitor routine is not finished this operation cycle, then the value is 1 otherwise the value is 0.

Bit5: This Bit is “testFailedSinceLastClear”. This bit notifies, monitor routine has reported that test has failed (at least once Bit0 is set) in any operation cycle at least once after clearing the DTC action is achieved. If the fault has happened after clear DTC is performed, then the value is 1 otherwise the value is 0.

Bit6: This Bit is “testNotCompletedThisOperationCycle”. This bit notifies that the monitor routine is still not running during this current operation cycle. This can be due to, the pin is not active for this operation cycle or when the request is sent from the tester, the monitor routine is not run. If the monitor routine is not run this operation cycle, then the value is 1 otherwise it is 0.

Bit7: This Bit is “warningIndicatorRequested”. This bit is used to draw the attention of the user or driver when the fault occurs. If fault occurs and any monitor is required for that exact fault, then the value is 1 otherwise the value is 0.

- **DTC CLASSES**

Class A DTCs: A class A code is a DTC that will result in the immediate illumination of the Malfunction Indicator Light. This type of code sets as a response for gross emission failure. For e.g., the misfire monitor can store a DTC and start flashing the MIL in response to its first recognition of a type A misfire. (A type A misfire is categorized as a severe misfire that could result in the overheating of the three-way catalytic converter, resulting in its damage)

Class B DTCs: Most DTCs in the engine control system are class B codes. A class B code states to a fault that does affect the vehicle’s emissions. When a fault related to an emissions are detected for the first time, a DTC for that fault is stored as a pending code. The Powertrain Control Module (PCM) does not light up the MIL at this time. During the next trip or drive cycle, the pending fault code will be erased only when the monitoring sequence that first identified the fault is repeated and the same fault does not repeat. If the fault does recur on the second trip or drive cycle, the pending code is then stored in memory as a confirmed code, also commonly denoted to as a mature code. It is at this point that the freeze frame data is stored and the MIL is illuminated by Powertrain Control Module (PCM).

Class C DTCs: A class C code is a DTC that defines a fault that does not adversely affect the vehicle’s emissions. Depending upon the vehicle, it may result in illumination of the MIL or “Service Engine Soon” light instead.

Class D DTCs: A class D code is a DTC that denotes to a fault that does not adversely affect the vehicle’s emissions and nor does it illuminate the MIL. These codes are the least important of the code types.

3.3 Off-Board diagnostics

Off-board diagnostics defines a systems outside the vehicle that can use the diagnostic services to read out data or start the execution of an on-board diagnostic test implemented as a part of an ECU. The Off-board diagnostics (UDS, KWP 2000, etc.) is typically some tool used on a computer in a repair shop or an end-of-line tester (tool that checks new-built vehicles at the end of the production line).
Off-board diagnostics can also be done on a server that is remotely connected to the vehicle, this is often called remote diagnostics and gives other possibilities to gather data and find faults. Remote diagnostics uses a diagnostic client that is employed in an ECU inside the vehicle and then this ECU is connected to an off-board server system which perform the diagnostic tasks, shown in Figure 13.

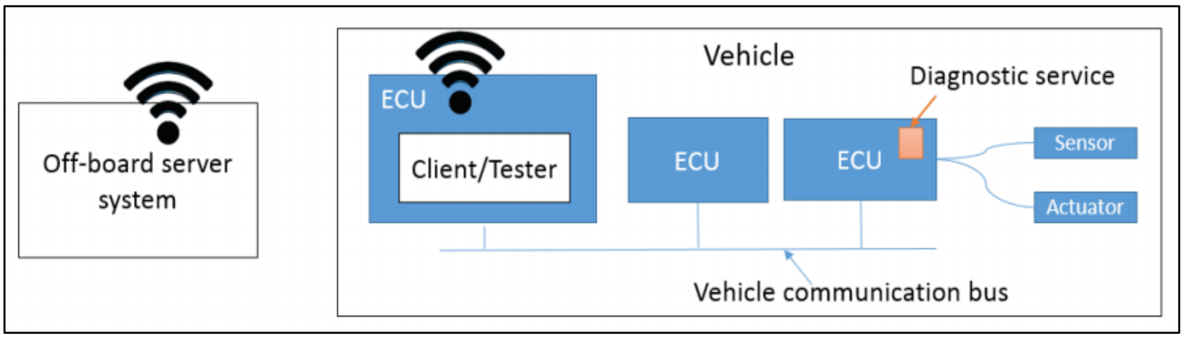


Fig. 13: Remote Vehicle diagnostics. Off-board server connected to a diagnostic tester/client in an ECU in the vehicle.

3.3.1 Unified Diagnostics Service (UDS)

Off-board vehicle diagnostics is used for the diagnostics of every other vehicle ECU function other than emission. There are several protocol standards defined for off-board diagnostics, however, Unified Diagnostics Services (UDS) [20] is the most popular diagnostic protocol. UDS (ISO 14229-1) is an International Standard that expands the individual properties which are different from data link layer requirements of an automotive diagnostic service in a road vehicle. It is based on the idea of Keyword Protocol (KWP2000) to fulfill common requirements for diagnostic systems on CAN buses. The UDS Protocol was created by merging the ISO Standards 14230-3 (KWP 2000) and 15765-3 (Diagnostics on CAN). This carried out to greatly decrease the costs which to date have arisen for the development of diagnostic communication. This standard provides a unified set of diagnostic services for ECUs.

There are five types of Diagnostics functions described in the specification as explained in table 3 below.

DIAGNOSTICS FUNCTIONS	**EXAMPLES**
Communication Management	Session Control, Device Reset, Security Access, Communication Control
Data	Read Identifiers or Memory Write Identifiers or Memory
Stored Data	Read Diagnostics Information Clear Diagnostics Information
I/O Control	Control Input or Output
Reprogramming	Download and Upload of Data

Table 3: UDS Diagnostics Functions

Basically it covers the implementation details of ISO 14229 services over CAN figure 14. The standard is based on Open Systems Interconnection (OSI). The services used by a diagnostic tester (client) and an ECU (server) are distinguished as: Unified diagnostic services (layer 7) and Communication services (layers 1 to 6).

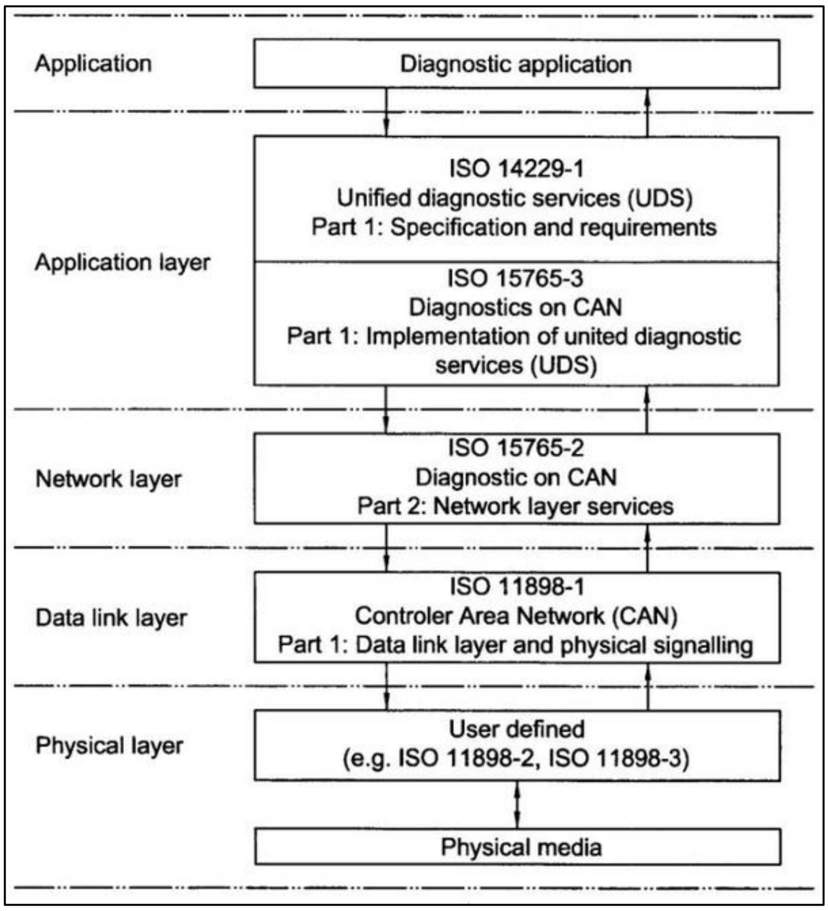

Application
Diagnostic application
Application layer
ISO 14229-1
Unified diagnostic services (UDS)
Part 1: Specification and requirements
ISO 15765-3
Diagnostics on CAN
Part 1: Implementation of united diagnostic services (UDS)
Network layer
ISO 15765-2
Diagnostic on CAN
Part 2: Network layer services
Data link layer
ISO 11898-1
Controler Area Network (CAN)
Part 1: Data link layer and physical signalling
Physical layer
User defined
(e.g. ISO 11898-2, ISO 11898-3)
Physical media

Fig. 14: Implementation of UDS protocol over CAN

3.3.2 UDS Request/Response

The main intension of UDS protocol is to communicate with all electronic data units that are positioned and interconnected in the vehicle, it also provide maintenance to check errors, actualizing the firmware, etc. In a diagnostic session, the network consist of tester (Client) and the ECU being tested (Server). A diagnostic service request is sent from the client to the server. The client starts with a service request and always ends with positive, negative or no response from the ECU (Figure 15). The transport protocol of UDS consists of ISO-TP [21]. ISOTP is an International Standard for transmitting data over the CAN bus which allows maximum data length up to 4095 bytes in a single data frame.

The three types of frames in UDS protocol.

1. Request Frame
2. Positive Response Frame
3. Negative Response Frame

Service ID – It is basically 1 byte ID belongs to the service well-defined in 14229-1. Server see this Identifier and perform that particular task related to this service.

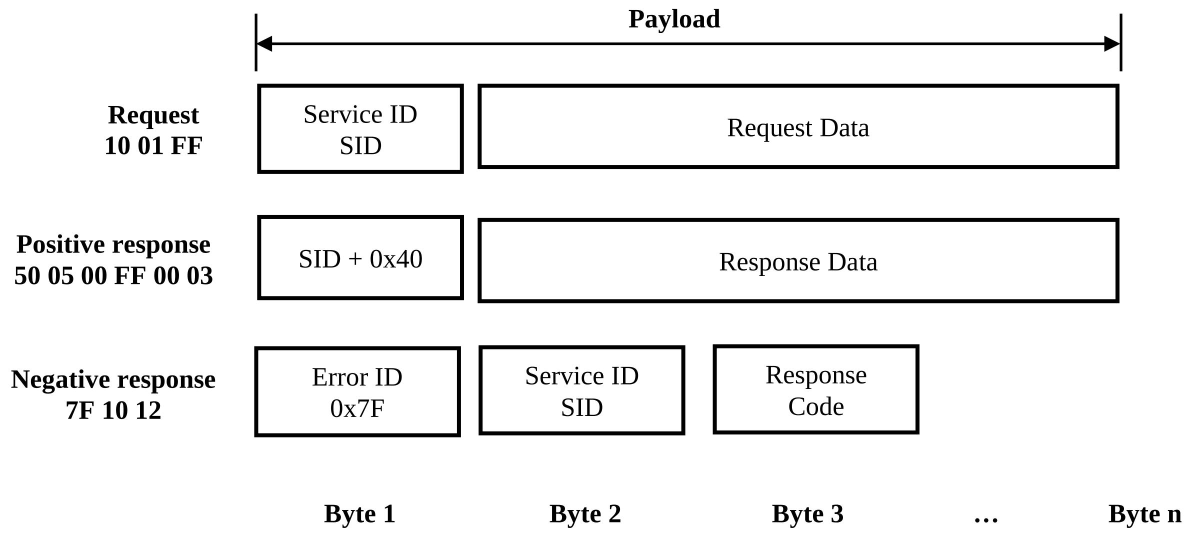


Fig. 15: UDS message format

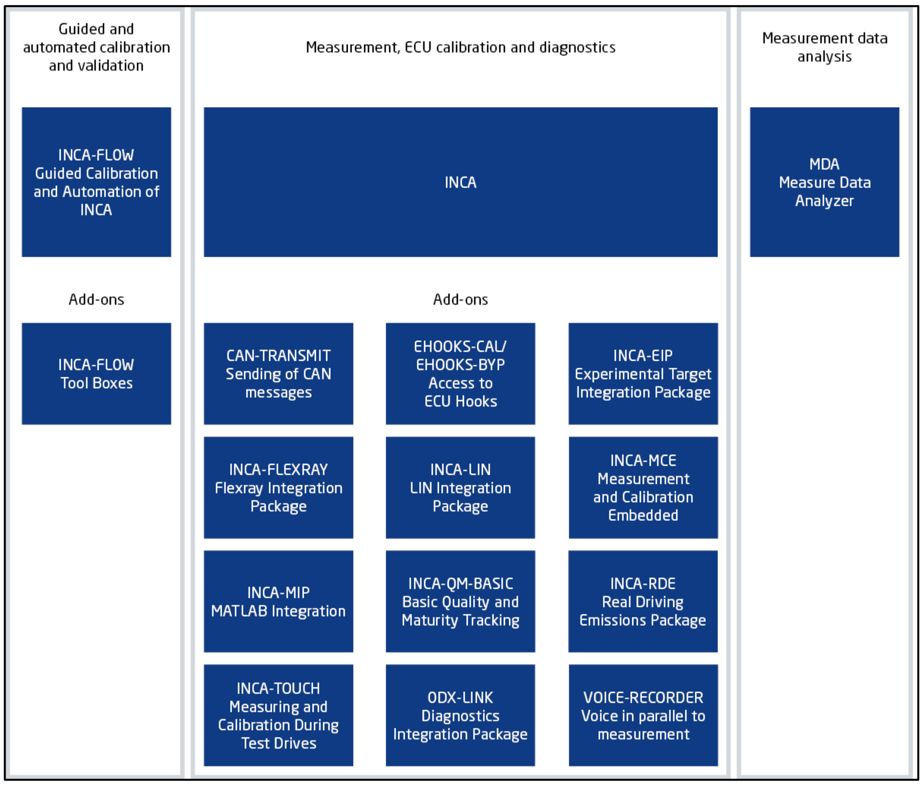
3.4 Diagnostic Management Software

An OBD II Powertrain control module (PCM) includes diagnostic management software to organize the complex testing procedures. The terms used for this diagnostic management software differ by manufacturer. In Porsche Engineering the most commonly used diagnostic software is INCA and DiagRA (which is used in this thesis work and explained in Section 5.1).

3.4.1 Integrated Calibration and Application Tool (INCA)

INCA is a measuring, calibration, and diagnostic system that provides wide-range of measuring support. INCA supports in all essential tasks during control unit calibration, evaluates the measured data, and documents the calibration results [22].

INCA can be used to read measured data from the control unit and the engine in parallel. This program helps to determine measured engine data such as lambda, different temperatures and voltage values, etc. INCA, is not just a tool that will adapt to a variety of different control units, but also a system that will optimize a wide range of different vehicle components.
It is an "open system". With consistent implementation of the ASAM-MCD standard and support for data exchange formats that are established in the environment allow this program to be used for any manufacturer's ECU interfaces and to be integrated in existing data processing infrastructures.


Guided and automated calibration and validation
Measurement, ECU calibration and diagnostics
Measurement data analysis
INCA-FLOW Guided Calibration and Automation of INCA
INCA
MDA Measure Data Analyzer
Add-ons
Add-ons
INCA-FLOW Tool Boxes
CAN-TRANSMIT Sending of CAN messages
EHOOKS-CAL/ EHOOKS-BYP Access to ECU Hooks
INCA-EIP Experimental Target Integration Package
INCA-FLEXRAY Flexray Integration Package
INCA-LIN LIN Integration Package
INCA-MCE Measurement and Calibration Embedded
INCA-MIP MATLAB Integration
INCA-QM-BASIC Basic Quality and Maturity Tracking
INCA-RDE Real Driving Emissions Package
INCA-TOUCH Measuring and Calibration During Test Drives
ODX-LINK Diagnostics Integration Package
VOICE-RECORDER Voice in parallel to measurement
Fig. 16: INCA System Overview

INCA consists of a measurement and calibration core system which can be enhanced by several add-ons and custom-made extensions (e.g. INCA-MIP, INCA-QM-BASIC, INCA-FLEXRAY) that can be integrated in INCA as shown in figure 16. In addition to that, INCA proposes open interfaces which allow for the adaptation of its core capabilities as well as the remote control of INCA by other applications.

INCA Measurement and ECU Calibration

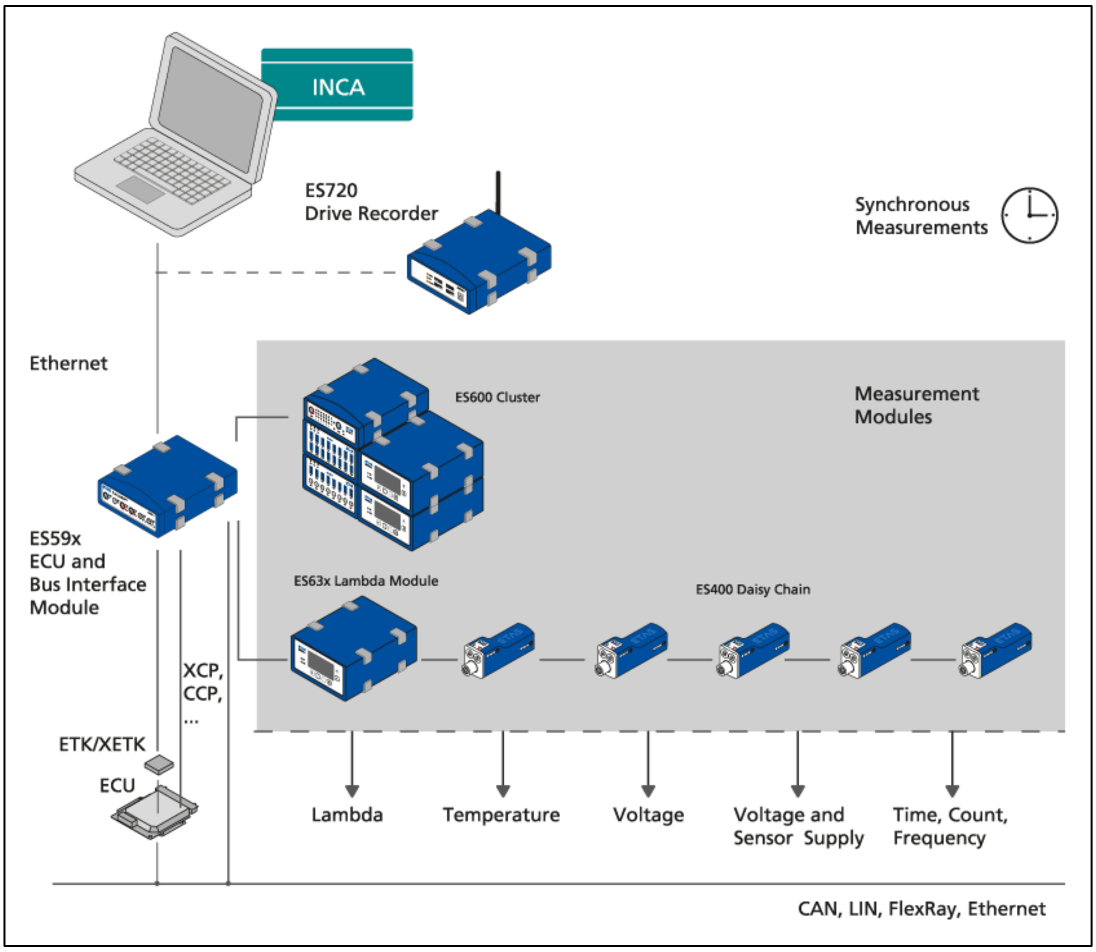


Fig. 17: INCA Interface for Measurement and ECU Calibration

It enables the adjustment of function parameters, maps, and tables either offline or during ECU runtime. This tool manages the ECU’s volatile and non-volatile data memory and resolves parameter dependencies. Using powerful editors present scalars, curves, or maps as tables or graphs in physical or hexadecimal format. Calibration scenarios consists of multiple parameter values of specific functions and ease the comparison of different settings.

For offline management of calibration data, it generates sophisticated functions for listing, comparing and merging datasets. In addition, INCA supports handling of meta-data describing the history and maturity of a parameter or function calibration with its Basic Quality and Maturity Tracking add-on.

In parallel to calibration, INCA provides for the acquisition of data from the ECU and vehicle buses such as CAN, LIN, Ethernet, and FlexRay as shown in figure 17. In addition, INCA measures various parameters from sensors and the vehicle environment. Quantities extracted from measurements and calibration variables can be calculated and displayed online. Using sophisticated trigger conditions data recording with several independent recorders may be started and stopped. Parallel recording of data associated with different trigger conditions is also possible. Data records comprises of the measured and calculated signals, calibration parameters, trigger options as well as user comments.

INCA Diagnostics

ODX-LINK tool adds ECU diagnostics capabilities to the measurement and calibration functionality of the INCA basic product. As the calibration and diagnostics related signals are acquired in parallel, therefore it can be used for triggering and calculation of derived signals in the same manner. All data is recorded in single measurement file and displayed in the same views. A single ECU and bus interface module can provide connections for both ECU diagnostics and calibration as shown in figure 18 below.

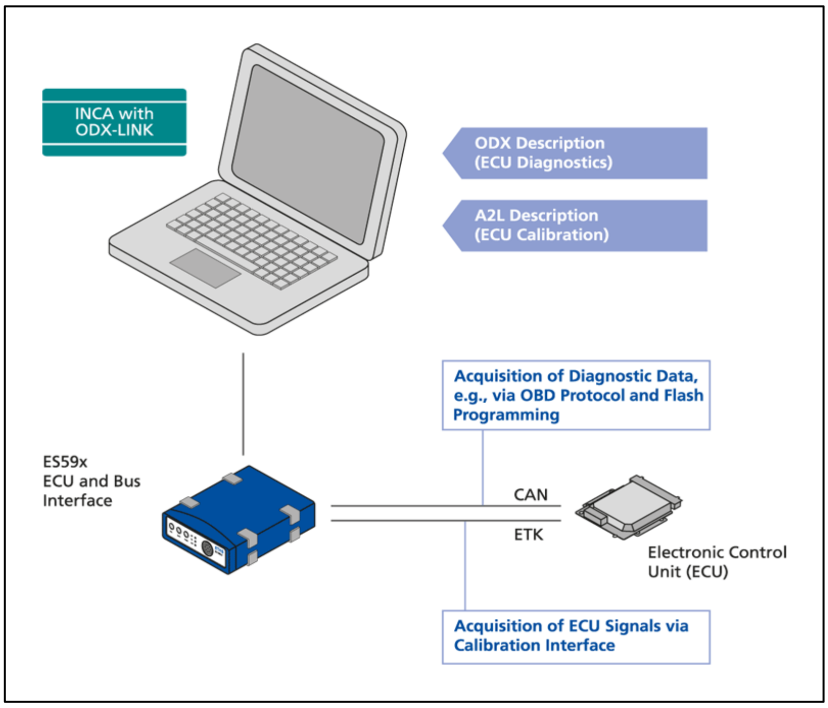

INCA with ODX-LINK
ODX Description (ECU Diagnostics)
A2L Description (ECU Calibration)
Acquisition of Diagnostic Data, e.g., via OBD Protocol and Flash Programming
ES59x ECU and Bus Interface
CAN
ETK
Electronic Control Unit (ECU)
Acquisition of ECU Signals via Calibration Interface

Fig. 18: INCA Interface with ODX-LINK

ODX-LINK integrates Scantool functions based on diagnostic services required by OBD emission regulations. Based on the services explained in ISO 15031-5 and SAE J1979, the easy to use OBD Scantool visualizes fault memory entries, status information of monitoring functions, vehicle information, in-use monitor performance ratios, and environmental data known as freeze frames.

Beyond OBD, ODX-LINK facilitates full diagnostics of ECUs compliant to the ODX standard (Open diagnostics data exchange). In addition, INCA can match a service tester and execute troubleshooting functions. Using this technique, service diagnostics can be validated long before service tester hardware is available. Using ODX-FLASH tool in INCA, a complete solution for validating ODX-based vehicle diagnostics and ECU reprogramming can be performed.

3.4.2 Important terms in Diagnostics

OBD II standards require that the engine management system should be able to detect faults, turn the MIL on or off, set DTCs in memory, and run drive cycles and trips for each monitored circuit according to the particular sets of operating conditions. Few of the important diagnostics concepts are explained further [23].

FREEZE FRAME DATA

Apart from storing detected DTCs, the diagnostic management software keeps a full record of all the relevant engine parameters for a given circuit.
If a fault is detected and logged, that information is stored as a snapshot. This data, known as freeze frame data, is used by the diagnostic management software for comparison and identification of comparable operating conditions when they recur. This data is used to provide further assistance in determining what might be a problem in the system. Also, this data can be used to help in duplicating the symptom during a road test. Freeze frame data can be retrieved with a Scantool through the data stream and typically includes the following:

- The DTC involved
- Engine RPM
- Engine load
- Fuel trim (short- and long-term)
- Engine coolant temperature
- MAP and/or MAF values
- Throttle position
- Operating mode (open or closed loop)
- Vehicle speed

On the basic system, freeze frame data store information only of the DTC that occurred first, unless a later DTC is of higher priority, such as a severe misfire or fuel system DTC. In this case, the diagnostic management software interchanges the stored data from the lower priority DTC with the freeze frame data related to the misfire or fuel system DTC.
According to the previous tests performed the freeze frame data which is recorded by the PCM starts recording after five seconds after it records the DTC in memory. As the driving conditions are measured during freeze frame, recording are most often the same as they were when the DTC was recorded. There is a small possibility for change during this five-second period, if the driver suddenly hit the brakes or hit the throttle to the floor.

WARNING LIGHTS (MIL AND EPC)

- Malfunction Indicator Light (MIL):

The MIL is also known as the Check Engine Light. The main purpose of this warning light is to indicate a detected problem and alert the driver about the issue with the vehicle.
The OBD II system turn on MIL when there is a problem with the vehicle engine, transmission or emission control system. There is always a reason if light turns so it’s always recommended not to ignore it and to investigate the cause. But it is totally normal for the light to illuminate for a few seconds after starting the engine and it should go out when the engine is running.

The Malfunction Indicator Light indicates three different types of problems:

1. Occasional flashes indicate temporary engine malfunctions. In this case, it is good to be aware of the probable forthcoming issues which can later on turn to more serious ones.
2. The most common case is when the indicator light stays on constantly. It indicates more serious problem that requires action to be taken as soon as possible. Yet, sometimes the issues aren’t that serious for example it can affect the emissions of the vehicle in a long run.
3. The most serious type of signaling is when the MIL flashes all the time. It is a sign that the engine is misfiring. This issue is very important and should stop the engine immediately to prevent serious damage. For instance, it can lead to overheating of the catalytic converter and even can cause fire.

OBD II scan tool is used to detect issue with the vehicle with the help of accompanied software through which it reads the Diagnostic Trouble Code(s) from the system. Every time the OBD system illuminate the MIL, it will also store a Diagnostic Trouble Code (DTC) in the electronic control unit. The OBD II system can turn the MIL off automatically if the conditions for the problem stop to exist. After checking a system or component for three consecutive times without spotting any problem, the light can be turned off, otherwise, usually it remains on.
With diagnostic software, the MIL reset is also possible. It is important to clear the MIL after fixing the problem. Because, for example, the vehicle will fail emissions testing if the MIL light is ON when tested.

- Electronic Power Control Light (EPC)

The EPC warning light is found in the instrument cluster and found on all Volkswagen, Audi, Seat and Skoda vehicles fitted with a drive-by-wire system. EPC stands for Electronic Power Control. This EPC light when illuminated displays the letters EPC to primarily warns the driver that there is problems in the engine's torque system (acceleration system).
If this light turn on, the vehicle’s throttle valve (butterfly) may be limited in order to protect the engine from damage. It avoids the engine rpm from revving above 2000rpm. This is known as limp mode and the ECU allows sufficient power to drive the vehicle to a service center for repairs. In order to remove this problem, the vehicle’s ECU should be scanned with an automotive diagnostic tool, in order to extract the DTC's related to the torque problem.

Mostly common problems cause this light to turn on is: the vehicle's knock sensors, its throttle system, its cruise control, its mass air flow system, its engine speed verification system or any of the other associated systems that cooperate in the drive-by-wire scheme. Few of them are explained as follow:

1. The Engine Speed Sensor is known to cause the EPC light to turn on. The engine speed sensor is a proximity magnetic transducer counting the rpm of the flywheel/crankshaft and sends a steady stream of pulses to the ECU. So when this stream of data is disturbed for whatsoever reason for only a fraction of a second, the ECU detects this and turns on the EPC light and cuts power to the engine. It does this to save the engine from damage. The following DTC error code, 17745 /P1337, 17746 / P1338, 17747 / P1339 and 17748 / P1340 are the troubles codes [24] which tells that the engine speed sensor is either loose or faulty.

2. The Accelerator Pedal is often also the main cause of EPC problems. The sign is that the engine idles a lot faster than it ought to. To verify this, physical pull the accelerator pedal away from the floor board while the vehicle is idling. If it reduces the engine's revs back to normal, then it is time to replace it, because the potentiometers that's built into the accelerator pedal, have gone faulty. Accelerator pedal problems are many and the following DTC, 16504/P0120, 16505/P0121, 16506/P0122, 16507/P0123, 18038/P1630, 18039/P1631, 18040/P1632, 18041/P1633, 18042/P1634, are linked to accelerator pedal errors.

3. Mass air flow sensor is another engine component that can cause the EPC light to turn on. Cleaning the mass air flow sensor with compressed air does often solve the problem but if the issue continues, then it’s time to replace mass air flow sensor. But before changing it, check to see if any of the rubber hoses in its surrounding area isn't perished. A leak in Air Intake System will permits unmonitored air to enter the intake which will throw a P2279 / 15093 error or a P0068/ 15101 error.

4. The Throttle body is by far the most common cause of an EPC problem though in many cases it is not the throttle body that's at fault but rather that the needs to be recalibrated (adaptation). The scan codes like P2135 / P2136 / P2137 / P2138 / P2139 and P2140, will give a good idea if the throttle body needs replacement. But in many cases it turns out to be the problem with wiring harness. The plugs that connects the throttle drive motor and the throttle position sensors is fairly troublesome and should be checked before throttle is replace.

5. The Brake light switch can also cause the EPC light to come on because the torque control circuit uses the brake light signal as an ECU input signal when the vehicle decelerates.

6. The Injectors and the Ignition Coils can also cause the EPC light to turn on. In case of spark plugs an incorrect gap can cause it while in injectors, driving with very less petrol in the tank can also cause the EPC light to turn on because the high pressure fuel pump may lose pressure which then informs the ECU to constrain the torque circuit and switch on the EPC light and make the vehicle go into limp mode.

OBD DRIVING CYCLE AND TRIP

- OBD Driving cycles

Warm-Up Cycle: OBD II standards define a warm-up cycle as a period of vehicle operation, after the engine is started in which coolant temperature rises by at least 4.4°C and reaches at least 71.1°C. Most OBD II DTCs are removed automatically after 40 warm-up cycles following the PCM turning off the MIL if the failure is not detected again.

Drive Cycle: A drive cycle is a series of operating conditions that allows the PCM to test all of the OBD II emissions-related monitors. When all of the driving conditions (known as enable criteria) have been met and all of the monitors have been run, the system is said to be inspection/ maintenance (I/M) equipped.
A drive cycle's enable criteria may be run in any order, but when the Powertrain control module sees all of the enable criteria performed that apply to a specific monitor, it will check off that monitor as being "complete." A scan tool can be used to regulate the status of each monitors, by observing which monitors have run successfully and which have not yet run and are still needed to complete the full drive cycle.
If the scan tool shows "N/A" for a monitor, that monitor is not applicable to that particular vehicle. If the Scantool indicates that the monitor has completed, this does not shows whether the monitor has passed or failed, only that the monitor's tests have been run. If the monitor has run and passed, there should be no stored DTCs in memory for that monitor; on the other hand, if the monitor has run and failed, there have to be a DTC stored in memory for that monitor.

- Trip

A trip is a diagnostic test that is designed to allow the PCM to determine a particular fault or DTC. It is generally run when a drive cycle has caused in the PCM setting a pending code in memory. A trip contains of a key cycle that includes ignition on, engine run, specific enable criteria met that let the PCM to run a diagnostic test, and ignition off long enough for the PCM to power down as shown in figure 19.

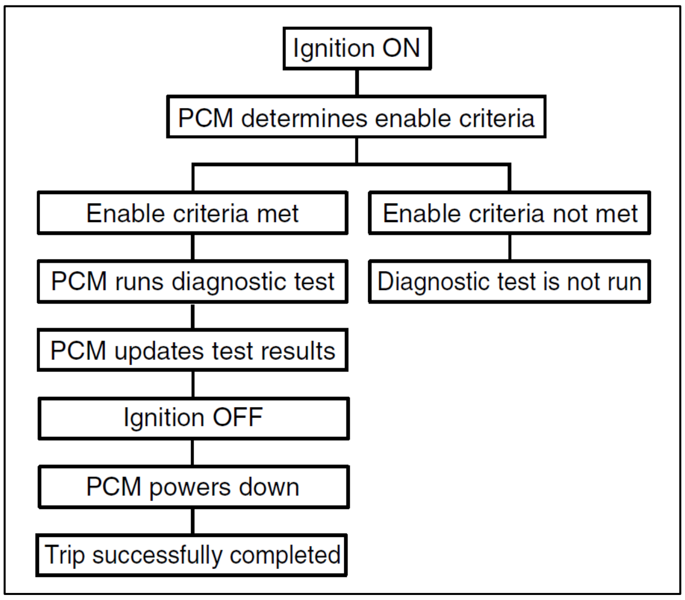


Fig. 19: OBD II Trip

A trip is used by the PCM to approve a pending code. If the fault is absent, then PCM will erase the pending code from its memory. If the fault is still present, the former pending code is turned into a confirmed code and the MIL is turned on. A trip can also be used by the PCM to confirm a repair after a DTC has been cleared from the PCM's memory with a scan tool or diagnostic software.

As every DTC and its related symptoms are unique, the enable criteria for the various DTCs are also unique to each DTC that the PCM may need to evaluate. Information is readily available that allows us to look up the specific enable criteria for any DTC.

Chapter 4
Hardware-in-the-Loop (HIL)

4.1 Introduction

Hardware-in-the-Loop simulation is one of the process used in the product development cycle in which one or more real components interact with components that are simulated in real time (dynamic models). The part of the system that is not simulated comprises of real devices, machines, or mechanical test benches. Nowadays, this term is mainly mentions to a real system which consists of one or more ECUs, controllers, or intelligent mechatronic modules for which a virtual environment is simulated electrically and dynamically. The simulated subsystem has to perform the following actions within one simulation step [25]:

- Read in the measurement signals (actuator control by the ECU)
- Calculate and perform numeric integration (simulate the entire dynamic model of a real system)
- Output the results (sensor simulation for the ECU).

The outcome is a closed loop between the real controller and the simulated plant. Failure to meet real-time conditions can result in unstable simulation and even damage of the real technical device. Figure 20 shows a signal flow that illustrates this structure.

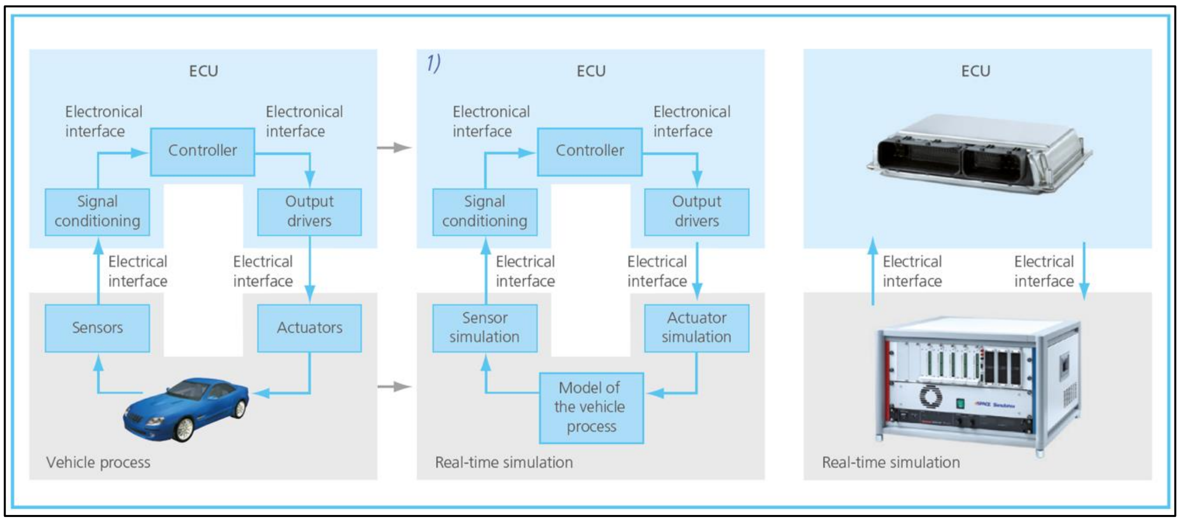


Fig. 20: Signal flows in a real system and in HIL [25]

4.2 V-cycle development process

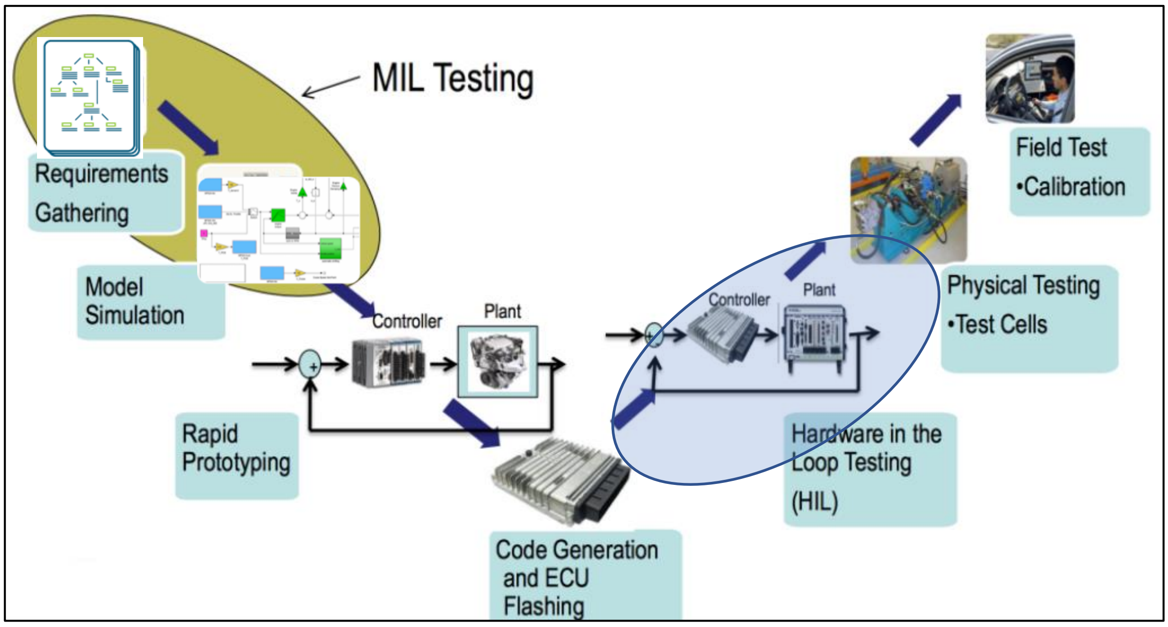


Fig. 21: V-Model of development process

Figure 21 illustrates the V-Cycle that defines a common control system development process.

- The process begins by defining a set of functional requirements for a system from which a simulation model of the control system has to be developed.
- The step following system specification is function specification, which can be supported by Model-In-the Loop simulation (MIL). Model-in-the loop simulation takes place on a PC with simulation models of both the specified functions as well as the vehicle. This step reports development of the functions as software models in graphically oriented programmer systems such as MATLAB/ Simulink.
- Following the development and testing of a simulation model, a Rapid Prototype phase comprises direct testing and optimization of the software models of the functions, with the appropriate software and hardware tools: on computer in the vehicle or on a test bench. MIL simulation and rapid prototyping assists in finding and eliminating any specification errors in an early phase.
- Next phase Software-In-Loop simulation (SIL). The software model used previously in model-in-the-loop simulation is replaced here by the later series code, and is incorporated into the simulation. This phase is typically performed by a supplier with specialist knowledge of a particular system.
- Following delivery of the ECU and software, HIL systems are used to carry out various forms of functional testing generally involves integration into the simulation loop of a number of real parts (actuators and sensors) e.g. a throttle-valve actuator.
- Physical test applications use transducer-based measurements (e.g. of temperature, pressure, stress / strain, sound, acceleration, etc.) to test the physical properties of the respective system components. Applications include the NVH test (noise, vibration,

harshness), which includes sound and vibration measurements from microphones and accelerometers.

- The final phase of the V-cycle is the use of vehicle testing for final verification of the software, hardware and its calibration. In this phase the vehicle is imitated on engine test benches – with transmission, driver, and driving resistance – in order to be able to “drive” e.g. WLTP cycle (world harmonized light-duty vehicles test cycle).

4.3 Porsche Engineering HIL Setup

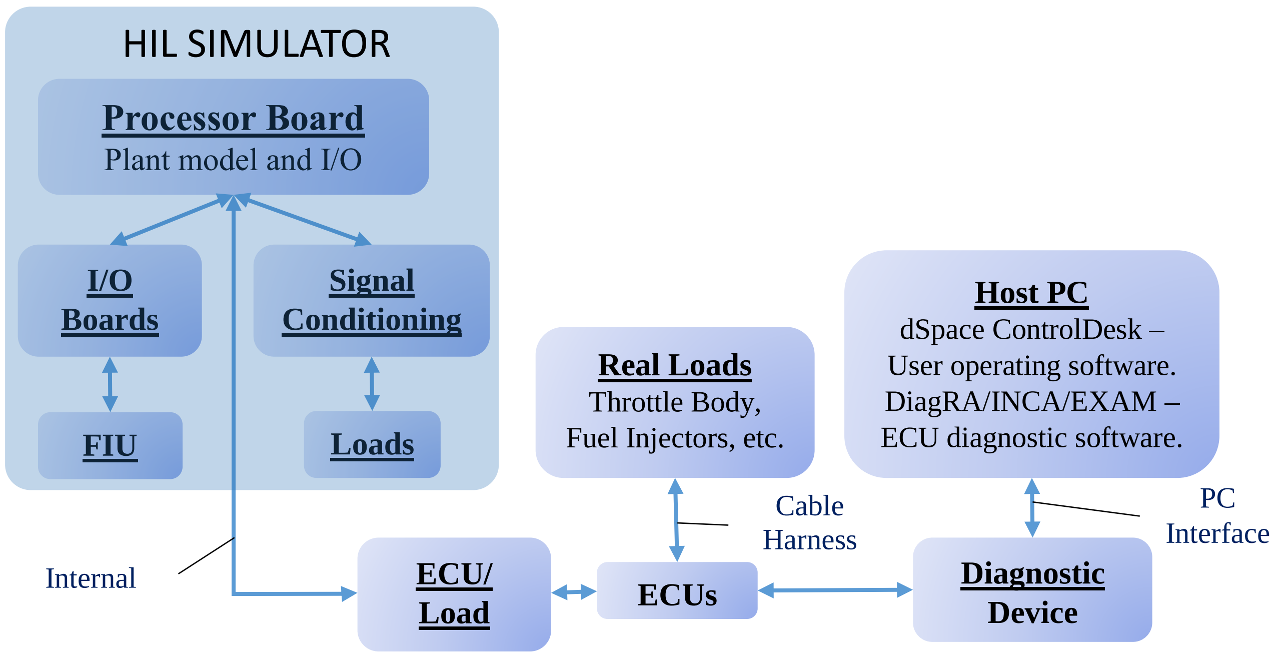


Fig. 22: Schematic setup of HIL System

The description of the HIL setup as shown in figure 22 is as follows [26]:

User Operating Software: Software which is used for the interactive interaction with the user. It is used to perform control and monitor tests, e.g. dSpace ControlDesk.

ECU Diagnostics Software: This software is used to read the error conditions detected by the ECUs connected to the HIL System. This software can also be used for the calibration id the ECU instead of diagnostics.

Diagnostic Device: A device which is used to provide access to the ECUs, it’s connected to the host PC for diagnostics or flash programming purposes.

HIL Simulator: Simulator substitutes the real environment of one or more ECUs by simulating the environment and its interaction with the ECUs in real-time in a closed loop.

Host PC: It is used to set up plant models, to build and to download real-time applications for the HIL Simulator. It consists of the operating and diagnostic software which is used to configure, control and analyze tests on the HIL Simulator, dSpace ControlDesk is shown in figure 23.

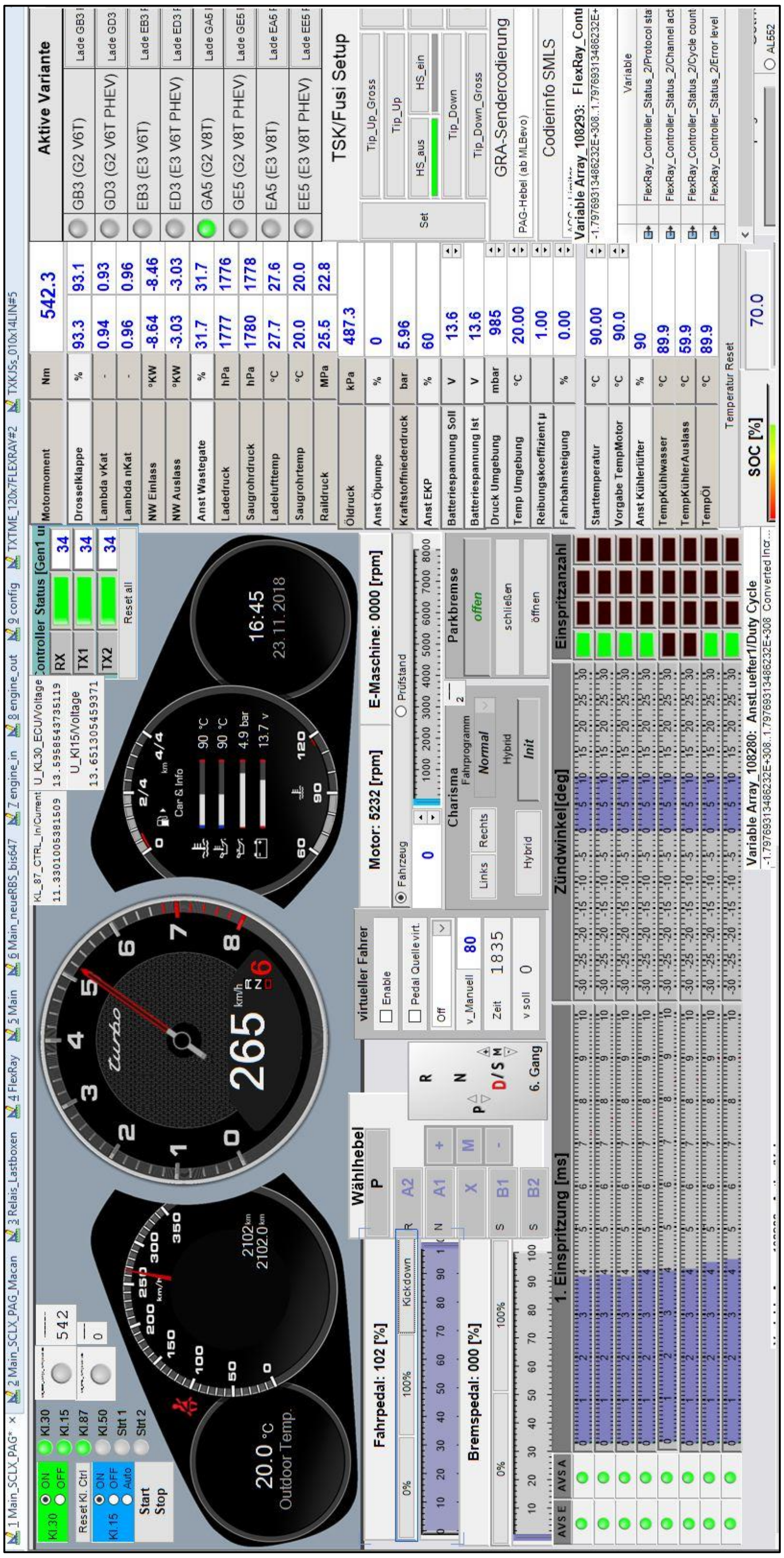


Fig. 23: dSpace Control Desk

Processor Board: It is a hardware component of the HIL Simulator for the real-time calculation of plant models and I/O models. Processor boards has the interfaces to the I/O boards in the HIL Simulator and to the host PC.

Plant Model: It is a model for real-time simulation of the system components (e.g. soft ECU, communication buses) and the systems environment (e.g. road model for test of an ESP ECU) that is not part the test as a real component. It enables closed loop operation of the ECU under test. A soft ECU is a real time model of an ECU that is required for the tests but not part of the tests as a real part. A restbus model is a reduced soft ECU variant, simulating bus communication.

I/O Model: Part of the real-time model with which the plant model input/output accesses the input/output physical channels of HIL system. The model for the restbus simulation is part of the I/O model.

I/O Boards: It is the hardware component which provides the HIL simulator Input/output channels. These channels are the interface to the input/output channels of the ECU to be tested.

Loads: ECU outputs are connected to the loads as a substitute for real loads that are part of the environment controlled by the ECU. These substitute loads enforce realistic currents on ECU outputs, which is a prerequisite for the proper operation of the connected ECUs, especially for diagnostics functions.

Internal Wiring: All wiring that are required to interconnect the hardware components within the HIL Simulator.
Signal Conditioning: It is also the hardware component which is used to match the signal level of the I/O boards to the signal levels of the connected ECUs.

Fault Insertion Unit (FIU): Component which generate electric faults on the inputs/outputs of the ECUs (actuators, sensors and bus channels). Typical error condition are, broken wire, short circuit to the ground, short circuit tot the supply voltage, short circuit between two terminals and loose contact.

ECU/Load Connector: One or more connection to the HIL system. The cable harness is connected to them in order to connect the HIL system to the ECUs under test.

ECU (Electronic Control Unit): One or more ECUs to be tested with the HIL system.

Real Loads: These are the real components which are used if they are tested in conjunction with the ECU.

Cable harness or external wiring: All wiring that is required to interconnect the HIL simulator with the ECUs and real loads.

Power Supply: Electrical circuits and connections for power supply of the HIL system, including connections to main supply it is not shown in the figure.

Practical Part

Chapter 5
Software Study

In this Chapter the software which are widely used in this work are discussed. DiagRA software is used to extract the relevant sensor or actuator data from the ECU using CAN-FD data-communication protocol. While to implement any driving cycle and generate it's TestCase Extended Automation Method (EXAM) is used.

5.1 DiagRA D – Diagnostic Software tool

5.1.1 Basics of the Software

DiagRA D is used worldwide by all major international automotive OEMs and 1st tier suppliers as a powerful diagnostics tool throughout all development stages. The functionality can be expanded by adding optional plugins.

The tool functions can be divided into three basic sections:

1. OEM specific workshop tester diagnostics – This function is a customer-specific part of the program, which is adapted by the different vehicle manufacturers. DiagRA can be used for all ECUs which are inside the vehicle.

2. Scan-Tool for OBDII/EOBD/HD-OBD/WWH-OBD diagnostics – The Scantool function is employed in compliance with SAE J1979 (OBDII/EOBD), SAE J1939 (HD-OBD) and ISO 27145 (WWHOBD). After the automatic configuration the supported data of all OBD significant ECUs is displayed. The SAE J1979 scan tool function supports all 10 services (Service \$01 – Service \$0A) as well as all the sub-functions (PIDs) defined by the market's relevant legislatures. The SAE J1939 scan tool function supports all essential diagnostic messages and parameter groups. WWH-OBD (World Wide Harmonized Onboard Diagnostics) is also involved according to ISO 27145 as part of this functional group and is complete for the vital diagnostic services including all DIDs (Data Identifier).

3. Advanced developer functions – The advanced developer functions are designed for automotive development engineers. By loading an A2L file the tool is able to read out and exhibit (depending on type of the fault memory manager): Display of Status-bits, detailed display of fault-code memory, reading memory locations and adaptation memory maps.

Further in this document, the workshop tester diagnostics function and the scan tool function will be discussed to make this study focused to the thesis work. And all the information in this chapter is from DiagRA user Manual [27].

LAYOUT OF THE MAIN WINDOW

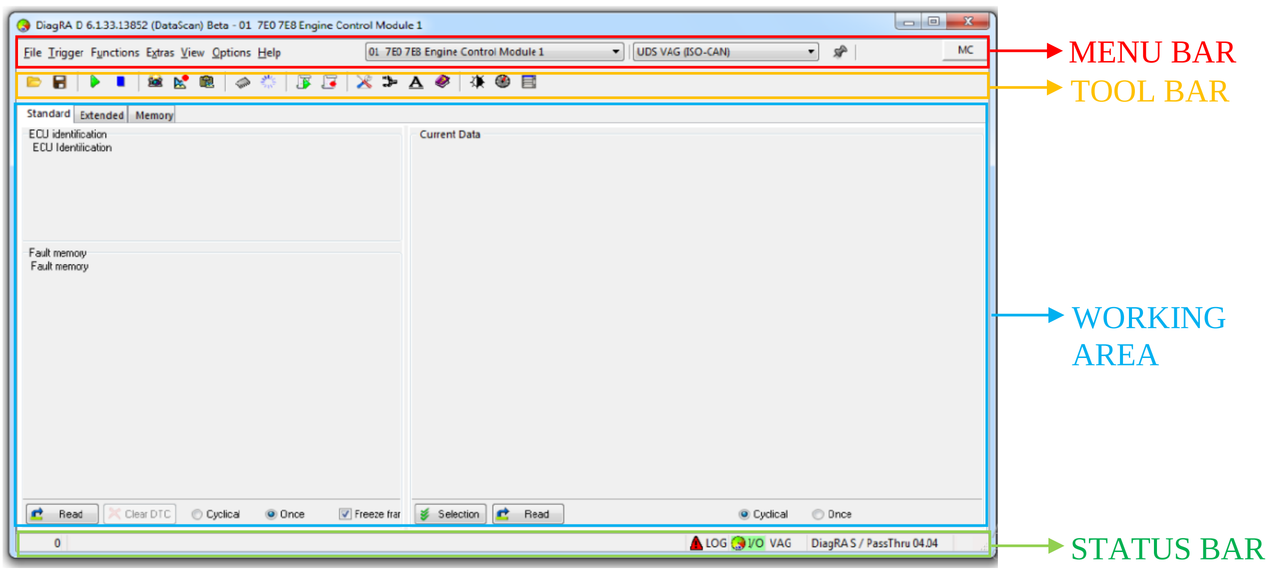


Fig. 24: Layout of the main window

MENU BAR - At the top, menu bar is to be found and right to it two drop-down lists. Here, the address word of the ECU is chosen, on which diagnostics has to be carried out, and the corresponding diagnostic protocol.

TOOL BAR - Right below the menu bar there is tool bar which allows access to the most important functions quickly via a single mouse-click.

STATUS BAR - At the bottom the status bar is situated. Here DiagRA D status information and basic settings is shown. In the far left there is a counter that runs from 0-255 during live communications to indicate each message exchanged between the PC and the ECU (This includes messages that form part of the communication structure). The counter provide feedback that the communication is running and serves as an indicator of the communication speed.
Next to it there is a textual description of what DiagRA D is doing at that particular moment: e.g. “Initializing Communication” or “Communication aborted”. To the right, the diagnostic data set and the diagnostic interface which are currently in used is to be found. Any loaded additional files, their names are shown in the status bars hint text.

WORKING AREA - The largest and most important area of the main window is the working area. It is divided into a series of tabsheets. Depending on the mode of application of DiagRA D at the current time that could be 10 or more. On the tab-sheets almost all functions that DiagRA D provides can be found.

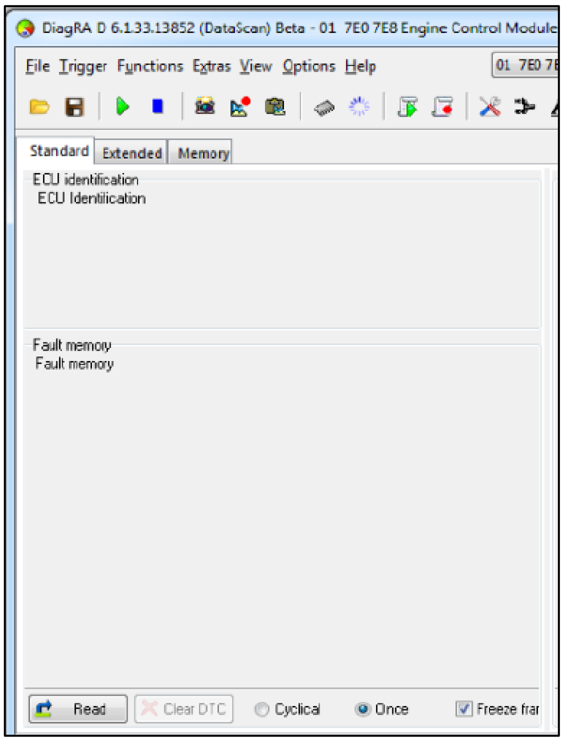
DiagRA D currently distinguishes about 30 different variations of communication protocols. The protocols differ in the physical communication layer (CAN, FlexRay, K-Line, Single-Wire-CAN or J1850), the transport layer (e.g. CAN TP 1.6, CAN TP 2.0, ISO-CAN) and often OEM specific variant of the topmost protocol layer (e.g. KWP 2000 5 Baud, KWP 2000 Fast Init, UDS). At Porsche Engineering the mostly used protocols are UDS VAG (ISO CAN). DaigRA D combines the possible options into a diagnostic protocol and makes them available for selection in a list.

5.1.2 DaigRA D as a Diagnostic tool

This section will discuss about which workshop tester functions have been implemented in DiagRA D. In the main window all of the tester-functions on the Standard, Extended and Memory tab-sheets are to be found. These functions do not need a description file.
The classical workshop diagnostics is a manufacturer specific often even ECU specific procedure and can have significant differences ranging from the implementation to the operation. The specific peculiarities are handled in DiagRA D by selecting the proper diagnostic data set and the corresponding diagnostic protocol. The parameterization, e.g. of fault codes, measurement values and identification data are handled by external files which are in turn organized into so called diagnostic data sets.

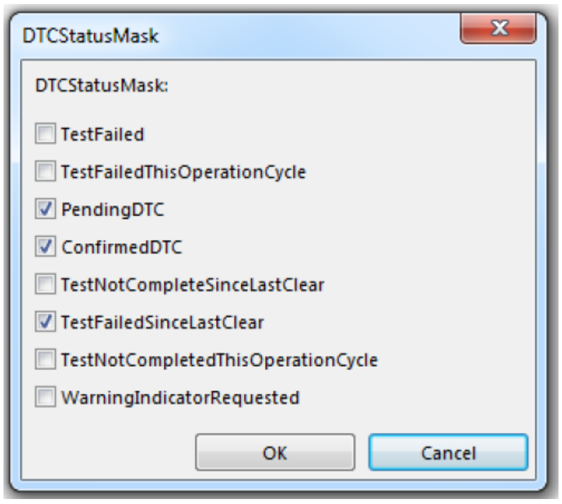
READ DTCs AND FREEZE FRAME DATA

Trouble codes are displayed on the Standard tab-sheet in the working area. This function is located on the left under Fault Code Memory. DTCs can be read cyclically and once "Cyclical" means that the DTCs will be read continuously, so once they have been read, they will be read again immediately thereafter. This selection is based on the requirements.



Cyclical [✓ Once]
✓ Freeze frame

(a)



DTCStatusMask:

- ☐ TestFailed
- ☐ TestFailedThisOperationCycle
- ✓ PendingDTC
- ✓ ConfirmedDTC
- ☐ TestNotCompleteSinceLastClear
- ✓ TestFailedSinceLastClear
- ☐ TestNotCompletedThisOperationCycle
- ☐ WarningIndicatorRequested

(b)

Fig .25: (a) Fault code memory (b)

If the ECU supports the reading of freeze-frame data, these will be shown at the same time. This function can be toggled on and off by the Freeze Frames button. The Clear DTC button is located next to the Read. For safety reasons, clearing the fault memory is only allowed after the fault memory was read once. If ECU allows to clear single trouble codes, the option will be provided with a dialog to make the selection.

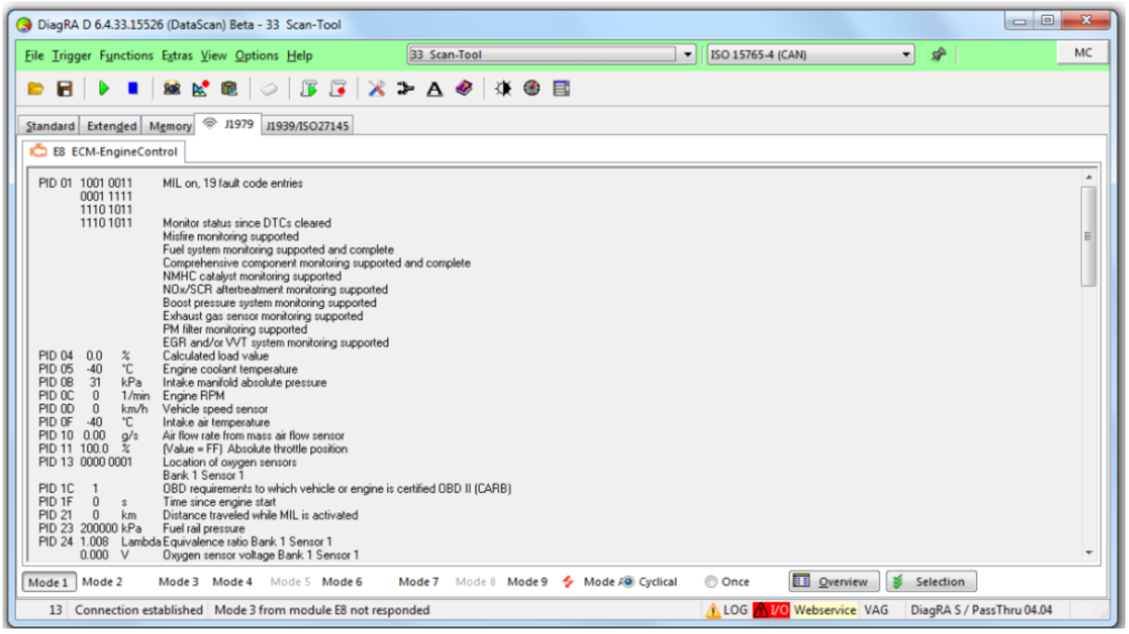
DTC status mask is set to select the DTC types which is to be read. To set this mask, right-click on the Read button and choose the option DTCStatusMask (Figure 25b). The following window will appear from which the DTC types which are relevant is selected. The selection will be carried out by clicking OK.

5.1.3 DaigRA D as a Scan tool

With DiagRA D reading of emission related information is done by using one of the OBD protocols or by using SAE J1939/ISO 27145. The address-word, which is used to initiate communication is 33 7DF 7E8 Scan-Tool. Using the SAE J1979 OBD II is used to measure real time data of vehicle which is further used in validating HIL.
When running Scan-Tool diagnostics, communication takes place simultaneously between all ECUs that support emission-related diagnostics. DiagRA D shows the measurement results of multiple ECUs at the same time. The upper area of the Scan-Tool tab-sheet is laid out for this purpose. The functionality of SAE J1979 is distributed on 10 functions, referred to as Mode 01 to Mode 0A (see also Section 3.2.2: Parameter identification numbers). Some of these functions have sub-functions, which may not be supported by all ECUs.

The functionality of SAE J1939 and SAE1979 is logically divided to different areas, e.g. Readiness, Freeze frames or Fault Codes. Switching between the different functionalities is done by clicking on the proper button at the bottom of the window.

SAE J1979


DiagRA D 6.4.33.15526 (DataScan) Beta - 33 Scan-Tool
File Trigger Functions Extras View Options Help
33 Scan-Tool
ISO 15765-4 (CAN)
MC
Standard Extended Memory J1979 J1939/ISO27145
E8 ECM-EngineControl
Mode 1 Mode 2 Mode 3 Mode 4 Mode 5 Mode 6 Mode 7 Mode 8 Mode 9 Mode A Cyclical Once Overview Selection

```
PID 01  1001 0011        MIL on, 19 fault code entries
        0001 1111
        1110 1011
        1110 1011        Monitor status since DTCs cleared
                         Misfire monitoring supported
                         Fuel system monitoring supported and complete
                         Comprehensive component monitoring supported and complete
                         NMHC catalyst monitoring supported
                         NOx/SCR aftertreatment monitoring supported
                         Boost pressure system monitoring supported
                         Exhaust gas sensor monitoring supported
                         PM filter monitoring supported
                         EGR and/or VVT system monitoring supported
PID 04    0.0    %       Calculated load value
PID 05    -40    °C      Engine coolant temperature
PID 0B    31     kPa     Intake manifold absolute pressure
PID 0C    0      1/min   Engine RPM
PID 0D    0      km/h    Vehicle speed sensor
PID 0F    -40    °C      Intake air temperature
PID 10   0.00    g/s     Air flow rate from mass air flow sensor
PID 11  100.0    %       (Value = FF) Absolute throttle position
PID 13  0000 0001        Location of oxygen sensors
                         Bank 1 Sensor 1
PID 1C    1              OBD requirements to which vehicle or engine is certified OBD II (CARB)
PID 1F    0      s       Time since engine start
PID 21    0      km      Distance traveled while MIL is activated
PID 23  200000   kPa     Fuel rail pressure
PID 24  1.008    Lambda  Equivalence ratio Bank 1 Sensor 1
        0.000    V       Oxygen sensor voltage Bank 1 Sensor 1
```

```
13 | Connection established | Mode 3 from module E8 not responded | LOG | I/O | Webservice | VAG | DiagRA S / PassThru 04.04
```

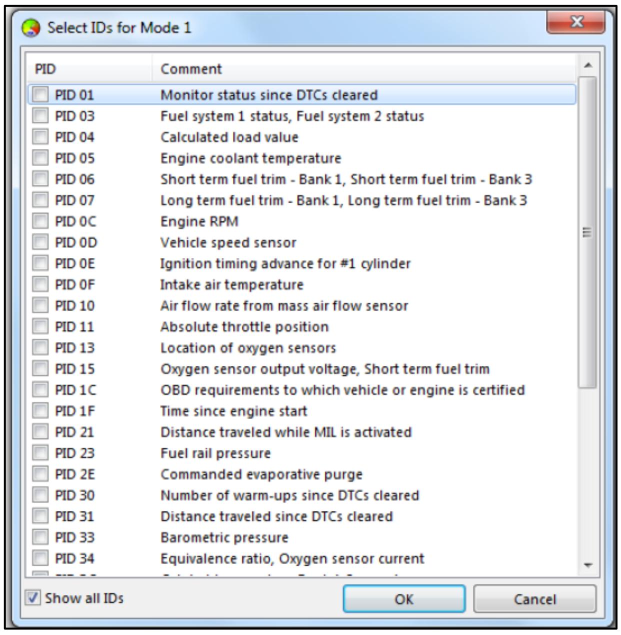
Fig. 26: The tab-sheet Scan-Tool

DiagRA D check which ECUs respond and which modes and sub-functions they support. Correspondingly, in the lower area of the J1979 tab, the buttons Mode 1 ... Mode A are laid out consecutively. Please note that Mode 4 (= Clear DTCs) is only made available after reading out the DTCs by calling either Mode 3, Mode 7 or Mode A. Also, each mode can be read (apart from Mode 4) cyclical as well as shown is figure 26.

SELECTION OF IDS (SAE 1979)

In order to keep the number of reported PIDs manageable, DiagRA D allows to select the IDs that should be displayed in Mode 1, 6 and 9.

1. Open the Select IDs for Mode X dialog (shown in figure 27) with Select button, where X is the current mode.


Select IDs for Mode 1

PID	Comment
PID 01	Monitor status since DTCs cleared
PID 03	Fuel system 1 status, Fuel system 2 status
PID 04	Calculated load value
PID 05	Engine coolant temperature
PID 06	Short term fuel trim - Bank 1, Short term fuel trim - Bank 3
PID 07	Long term fuel trim - Bank 1, Long term fuel trim - Bank 3
PID 0C	Engine RPM
PID 0D	Vehicle speed sensor
PID 0E	Ignition timing advance for #1 cylinder
PID 0F	Intake air temperature
PID 10	Air flow rate from mass air flow sensor
PID 11	Absolute throttle position
PID 13	Location of oxygen sensors
PID 15	Oxygen sensor output voltage, Short term fuel trim
PID 1C	OBD requirements to which vehicle or engine is certified
PID 1F	Time since engine start
PID 21	Distance traveled while MIL is activated
PID 23	Fuel rail pressure
PID 2E	Commanded evaporative purge
PID 30	Number of warm-ups since DTCs cleared
PID 31	Distance traveled since DTCs cleared
PID 33	Barometric pressure
PID 34	Equivalence ratio, Oxygen sensor current

Show all IDs [✓]

Fig. 27: Selection of IDs (SAE 1979)

2. Using the button Show all IDs, toggle between the desired selection and all IDs. If no IDs are selected here, all IDs will be displayed.

5.2 Extended Automation Method (EXAM)

EXAM (EXtented Automation Method) is a test management system software (in other words-automation tool) used by Porsche AG (parent company) in their large HIL system to test complete electronic vehicle functionality. This thesis conducted at The Porsche Engineering Services also utilizes EXAM distributed by MicroNova AG, commissioned by the VW-concern as test management system. WLTP cycle is implemented using TestCase generated by EXAM automation tool. Detailed discussion is in next chapter.

EXAM is built on Java and couples a ’drag-and-drop’ graphical user interface (GUI) onto HiL or SiL test systems. This test management system uses Python as programming language. The test case descriptions are written in Rational DOORS (a requirement program built by IBM). These requirements have to be synchronized with EXAM and specific test cases can then be created in EXAM regarding these requirements [28]. Any EXAM test process management is performed in a specific steps as shown in figure 28

The three most important views in EXAM are the Modeler explorer, the Testrunner perspective and the Reportmanager perspective.

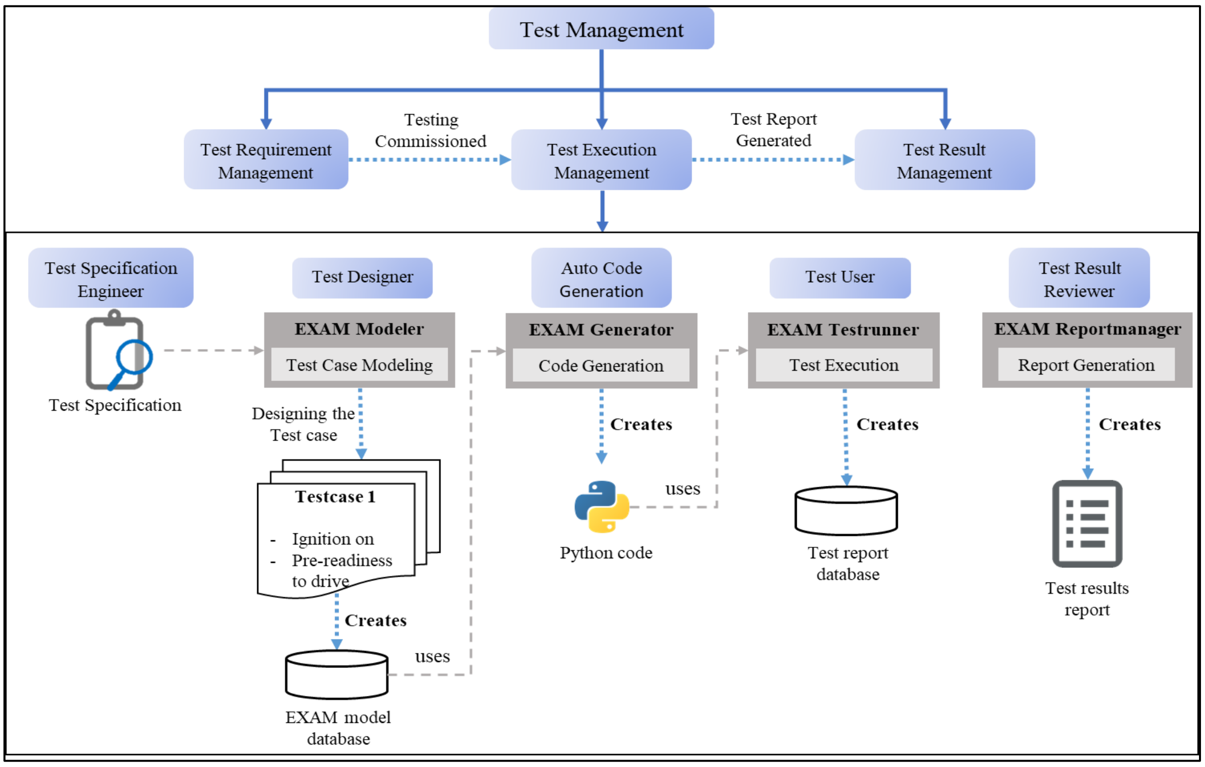
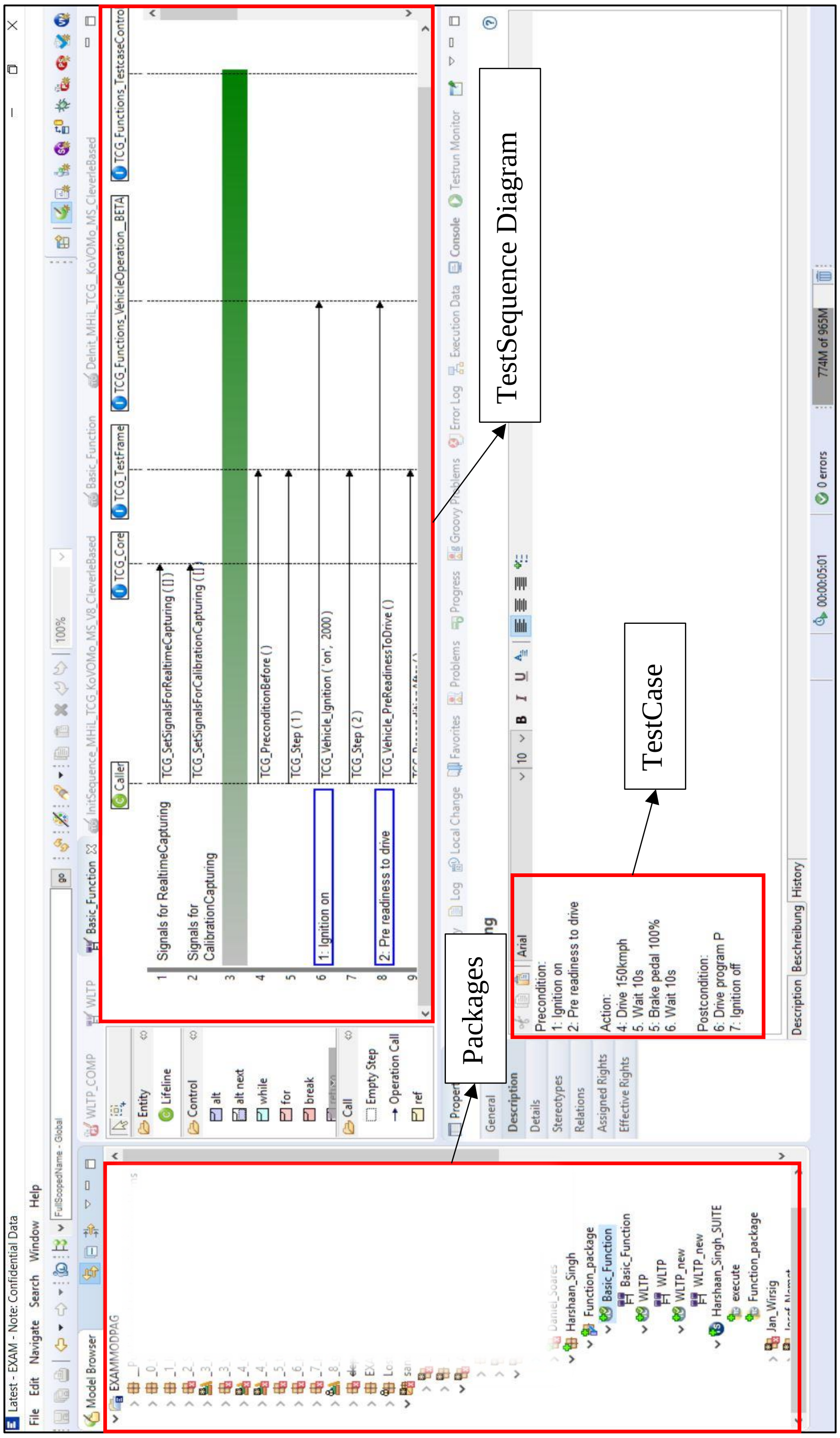


Fig. 28: EXAM Test Process

5.2.1 Modeler Perspective

In the modeler perspective, the tree structure shows how the system is built out of all the available functions. This tree structure consists of packages with unique names. In these packages classes are created with functions and parameters. These are the functions that can be used during the creation of test cases as shown in figure 29. To be able to reuse functions of the Testhaus and vice-versa, the whole tree structure of the Testhaus testing department is included in the workspace of EXAM.



Precondition:
1: Ignition on
2: Pre readiness to drive

Action:
4: Drive 150kmph
5. Wait 10s
5: Brake pedal 100%
6. Wait 10s

Postcondition:
6: Drive program P
7: Ignition off

Fig. 29: EXAM modeler perspective

Modeler perspective in general:

- Test case modeling and implementation
- User defined filters can be applied to the model browser
- Object compositions are automatically generated as far as possible
- Object properties, descriptions and relations as well as project properties are available from views (tab) and generated python code for object is also accessible.

5.2.2 Testrunner Perspective

The second window is the testrunner perspective as shown in figure 30. To understand this perspective, the definition of test suite is very important. TestSuite defines a self-contained test topic and its content. It is organized by TestGroups (the order in which TestCases and AdministrativeCases are executed). When test cases are coupled together with test suites, they can be selected to run automatically in EXAM. Together with the test suites, the right configuration of the test suite, where all the right classes are coupled onto the interfaces, are loaded when executing a test suite.

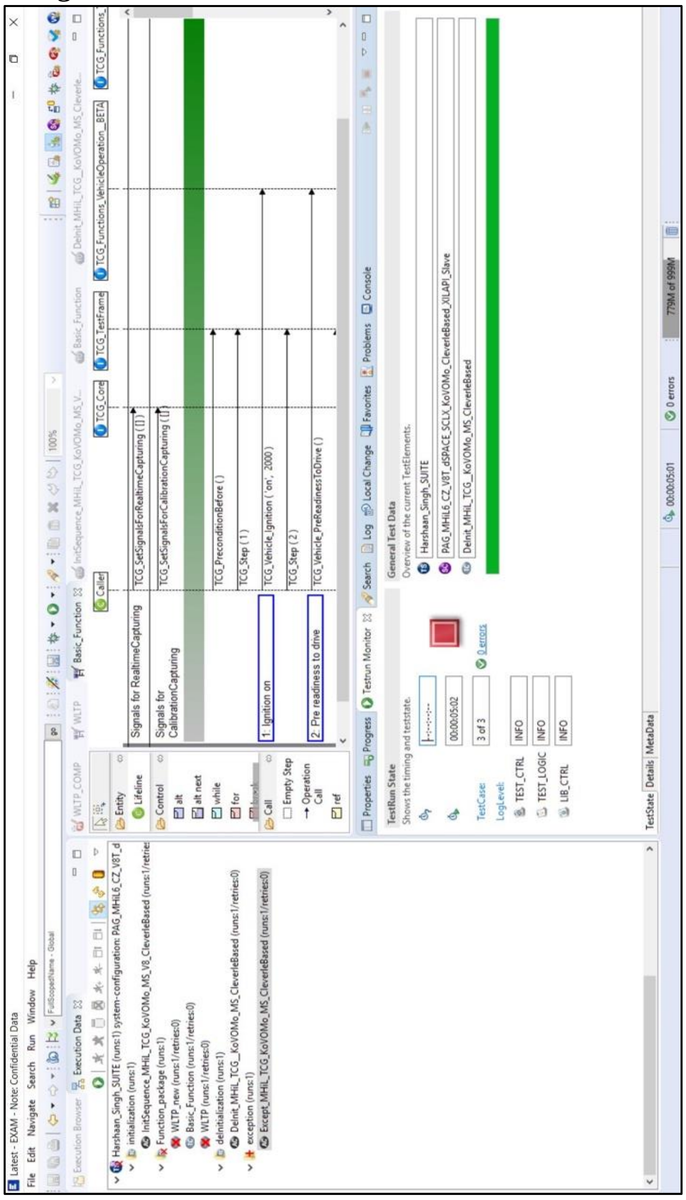
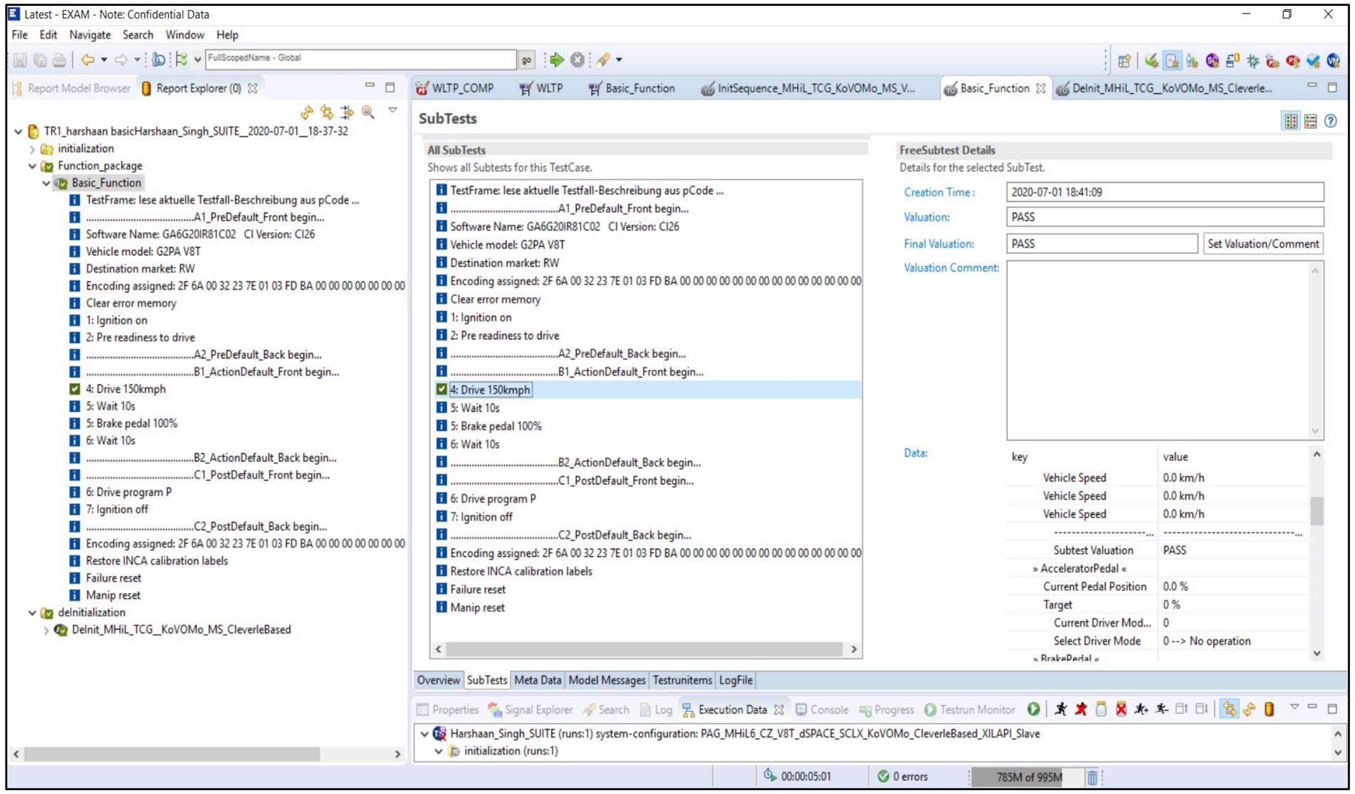

Latest - EXAM - Note: Confidential Data
File Edit Navigate Search Run Window Help
Execution Browser
Execution Data
Harshaan_Singh_SUITE (runs:1) system-configuration: PAG_MHIL6_CZ_V8T_d
initialization (runs:1)
InitSequence_MHIL_TCG_KoVOMo_MS_V8_CleverleBased (runs:1/retries
Function_package (runs:1)
WLTP_new (runs:1/retries:0)
Basic_Function (runs:1/retries:0)
WLTP (runs:1/retries:0)
deinitialization (runs:1)
Deinit_MHIL_TCG_KoVOMo_MS_CleverleBased (runs:1/retries:0)
exception (runs:1)
Except_MHIL_TCG_KoVOMo_MS_CleverleBased (runs:1/retries:0)
WLTP_COMP
WLTP
Basic_Function
InitSequence_MHIL_TCG_KoVOMo_MS_V...
Basic_Function
Deinit_MHIL_TCG_KoVOMo_MS_Cleverle...
Entity
Lifeline
Control
alt
alt next
while
for
Call
Empty Step
Operation Call
ref
Caller
TCG_Core
TCG_TestFrame
TCG_Functions_VehicleOperation_BETA
TCG_Functions
Signals for RealtimeCapturing
Signals for CalibrationCapturing
1. Ignition on
2. Pre readiness to drive
TCG_SetSignalsForRealtimeCapturing ([])
TCG_SetSignalsForCalibrationCapturing ([])
TCG_PreconditionBefore ( )
TCG_Step (1)
TCG_Vehicle_Ignition ('on', 2000)
TCG_Step (2)
TCG_Vehicle_PreReadinessToDrive ( )
Properties
Progress
Testrun Monitor
Search
Log
Local Change
Favorites
Problems
Console
TestRun State
Shows the timing and teststate.
00:00:05:02
3 of 3
0 errors
TestCase:
LogLevel:
TEST_CTRL INFO
TEST_LOGIC INFO
LIB_CTRL INFO
General Test Data
Overview of the current TestElements.
Harshaan_Singh_SUITE
PAG_MHIL6_CZ_V8T_dSPACE_SCLX_KoVOMo_CleverleBased_XILAPI_Slave
Deinit_MHIL_TCG_KoVOMo_MS_CleverleBased
TestState
Details
MetaData
00:00:05:01
0 errors
779M of 999M

Fig. 30: EXAM Testrunner perspective

Testrunner perspective in general:

- Run TestSuites
- Open and save individual test runs in an execution file with python code path (XML)
- Monitor and control runs in the testrun monitor view and console
- Property to set alterative python code path for test run
- View test run duration and estimated time
- Record all test case relevant data and offline reply of test cases with relevant data for additional analysis to save eg. HIL usage time

5.2.3 Reportmanager Perspective



Latest - EXAM - Note: Confidential Data
File Edit Navigate Search Window Help
FullScopedName - Global
Report Model Browser
Report Explorer (0)
WLTP_COMP
WLTP
Basic_Function
InitSequence_MHiL_TCG_KoVOMo_MS_V...
Basic_Function
DeInit_MHiL_TCG__KoVOMo_MS_Cleverle...
Overview SubTests Meta Data Model Messages Testrunitems LogFile
Properties Signal Explorer Search Log Execution Data Console Progress Testrun Monitor

```
TR1_harshaan basicHarshaan_Singh_SUITE__2020-07-01__18-37-32
  initialization
  Function_package
    Basic_Function
      TestFrame: lese aktuelle Testfall-Beschreibung aus pCode ...
      ......................................A1_PreDefault_Front begin...
      Software Name: GA6G20IR81C02   CI Version: CI26
      Vehicle model: G2PA V8T
      Destination market: RW
      Encoding assigned: 2F 6A 00 32 23 7E 01 03 FD BA 00 00 00 00 00 00 00
      Clear error memory
      1: Ignition on
      2: Pre readiness to drive
      ......................................A2_PreDefault_Back begin...
      ......................................B1_ActionDefault_Front begin...
      4: Drive 150kmph
      5: Wait 10s
      5: Brake pedal 100%
      6: Wait 10s
      ......................................B2_ActionDefault_Back begin...
      ......................................C1_PostDefault_Front begin...
      6: Drive program P
      7: Ignition off
      ......................................C2_PostDefault_Back begin...
      Encoding assigned: 2F 6A 00 32 23 7E 01 03 FD BA 00 00 00 00 00 00 00
      Restore INCA calibration labels
      Failure reset
      Manip reset
  deInitialization
    DeInit_MHiL_TCG__KoVOMo_MS_CleverleBased
```

SubTests

All SubTests
Shows all Subtests for this TestCase.

```
TestFrame: lese aktuelle Testfall-Beschreibung aus pCode ...
......................................A1_PreDefault_Front begin...
Software Name: GA6G20IR81C02   CI Version: CI26
Vehicle model: G2PA V8T
Destination market: RW
Encoding assigned: 2F 6A 00 32 23 7E 01 03 FD BA 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Clear error memory
1: Ignition on
2: Pre readiness to drive
......................................A2_PreDefault_Back begin...
......................................B1_ActionDefault_Front begin...
4: Drive 150kmph
5: Wait 10s
5: Brake pedal 100%
6: Wait 10s
......................................B2_ActionDefault_Back begin...
......................................C1_PostDefault_Front begin...
6: Drive program P
7: Ignition off
......................................C2_PostDefault_Back begin...
Encoding assigned: 2F 6A 00 32 23 7E 01 03 FD BA 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Restore INCA calibration labels
Failure reset
Manip reset
```

FreeSubtest Details
Details for the selected SubTest.

Creation Time :	2020-07-01 18:41:09
Valuation:	PASS
Final Valuation:	PASS [Set Valuation/Comment]
Valuation Comment:	

Data:

key	value
Vehicle Speed	0.0 km/h
Vehicle Speed	0.0 km/h
Vehicle Speed	0.0 km/h
------------------------...	---------------------------------...
Subtest Valuation	PASS
» AcceleratorPedal «	
Current Pedal Position	0.0 %
Target	0 %
Current Driver Mod...	0
Select Driver Mode	0 --> No operation
» BrakePedal «	

```
Harshaan_Singh_SUITE (runs:1) system-configuration: PAG_MHiL6_CZ_V8T_dSPACE_SCLX_KoVOMo_CleverleBased_XILAPI_Slave
  initialization (runs:1)
```

00:00:05:01   0 errors   785M of 995M

Fig. 31: EXAM Reportmanager perspective

After the execution of the test suite, a report will be generated by EXAM where the failed and successful test cases are listed as shown in figure 31. If the test cases are written correctly, the reason in case of failure of the test case will be shown in the reports. The reports are sent to a specific folder on the server where it is managed, analyzed and processed in the reportmanager perspective of EXAM.

Reportmanager perspective in general:

- Report data can be saved in multiple report databases as well as simultaneously managed
- Test results are structured into sub-tests to allow an efficient test analysis
- Some report elements can be edited in order to correct and comment on test results
- Report can be exported to XML and PDF files
- OpenSource BIRT framework handles PDF exports, featuring custom report templates.

Chapter 6
Worldwide Harmonized Light Vehicles Test Procedure (WLTP)

WLTP cycle (world harmonized light-duty vehicles test procedure) is a worldwide, harmonized standard used for estimation of the levels of pollutants, CO_2 emissions and fuel consumption of conventional, hybrid and electric vehicles. This new protocol was developed by the United Nations Economic Commission for Europe (UNECE) aiming to replace the new European driving cycle (NEDC) as the European vehicle homologation procedure. Its final class 3 version 5.3 was released in 2015 which is further used for the study as shown in figure 32 [29].

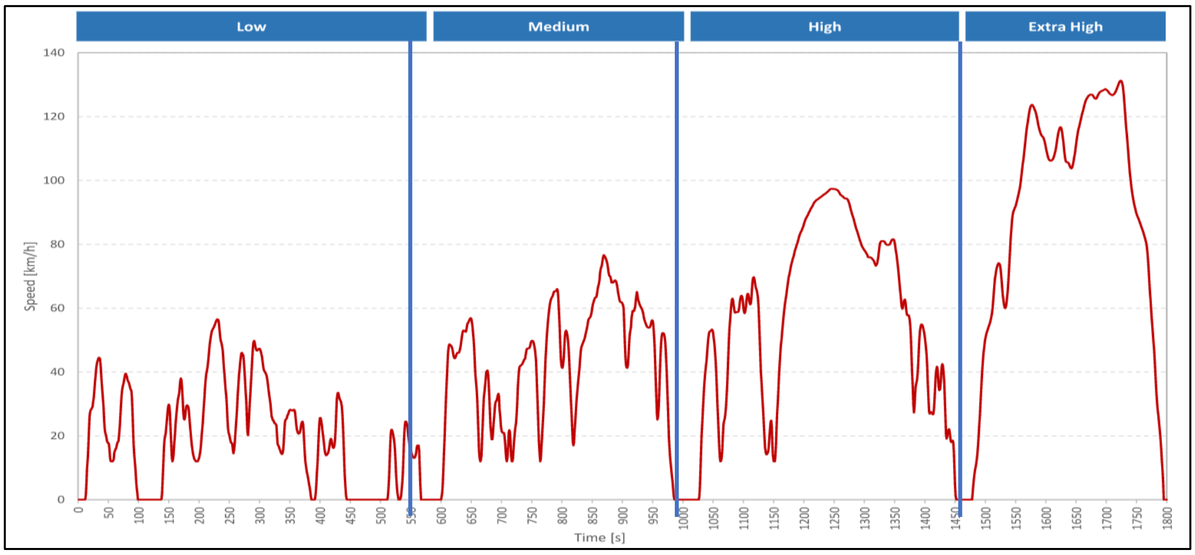


Fig. 32: WLTP class 3 v5.3 driving

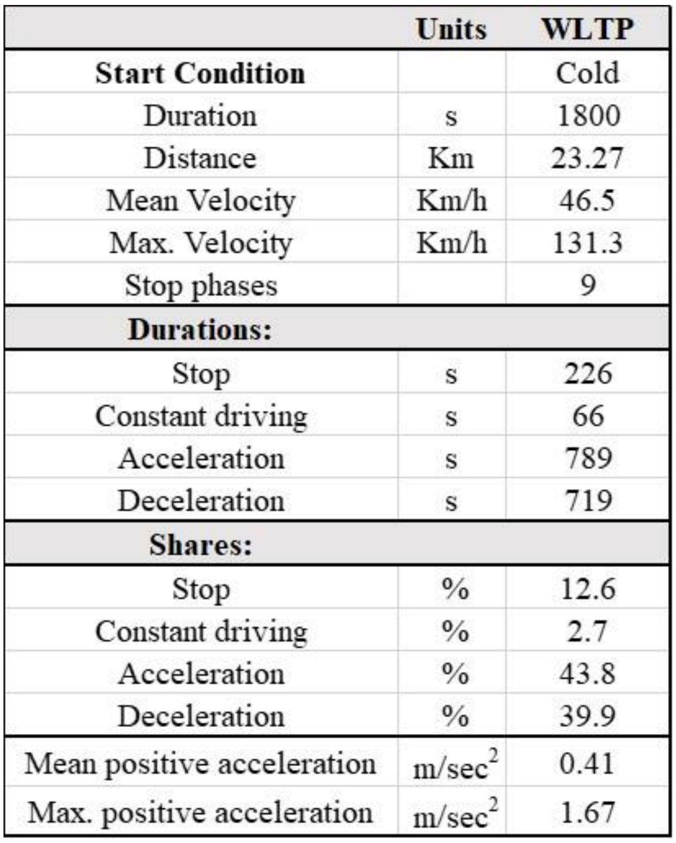
This test procedure is responsible for a very strict regulation regarding dynamometer tests and road load (motion resistance), gear shifting, total vehicle weight (by including optional equipment, load and passengers), fuel quality, ambient temperature, and tire selection and pressure.

Three different WLTP cycles are applied, depending on vehicle class differentiated by power/weight ratio PW in W/kg (rated engine power/curb weight): Class 1 – low power vehicles with $PW \leq 22$, Class 2 – vehicles with $22 < PW \leq 34$ and Class 3 – high-power vehicles with $PW > 34$. In this study class 3 test cycle is used as the Porsche KOVOMO V6 and V8 engines PW is more than 150 W/kg.

Also, the WLTP is divided into 4 different sub-parts shown in figure 32, each one with a different maximum speed:

- Low, up to 56.5 km/h
- Medium, up to 76.6 km/h
- High, up to 97.4 km/h
- Extra-high, up to 131.3 km/h

These driving phases mimic urban, sub-urban, rural and highway situations respectively, with an equal separation between urban and non-urban paths (52% and 48%). Table 4 quantifies the main descriptive parameters of the driving cycle.



	Units	WLTP
Start Condition		Cold
Duration	s	1800
Distance	Km	23.27
Mean Velocity	Km/h	46.5
Max. Velocity	Km/h	131.3
Stop phases		9
Durations:		
Stop	s	226
Constant driving	s	66
Acceleration	s	789
Deceleration	s	719
Shares:		
Stop	%	12.6
Constant driving	%	2.7
Acceleration	%	43.8
Deceleration	%	39.9
Mean positive acceleration	m/sec^2	0.41
Max. positive acceleration	m/sec^2	1.67

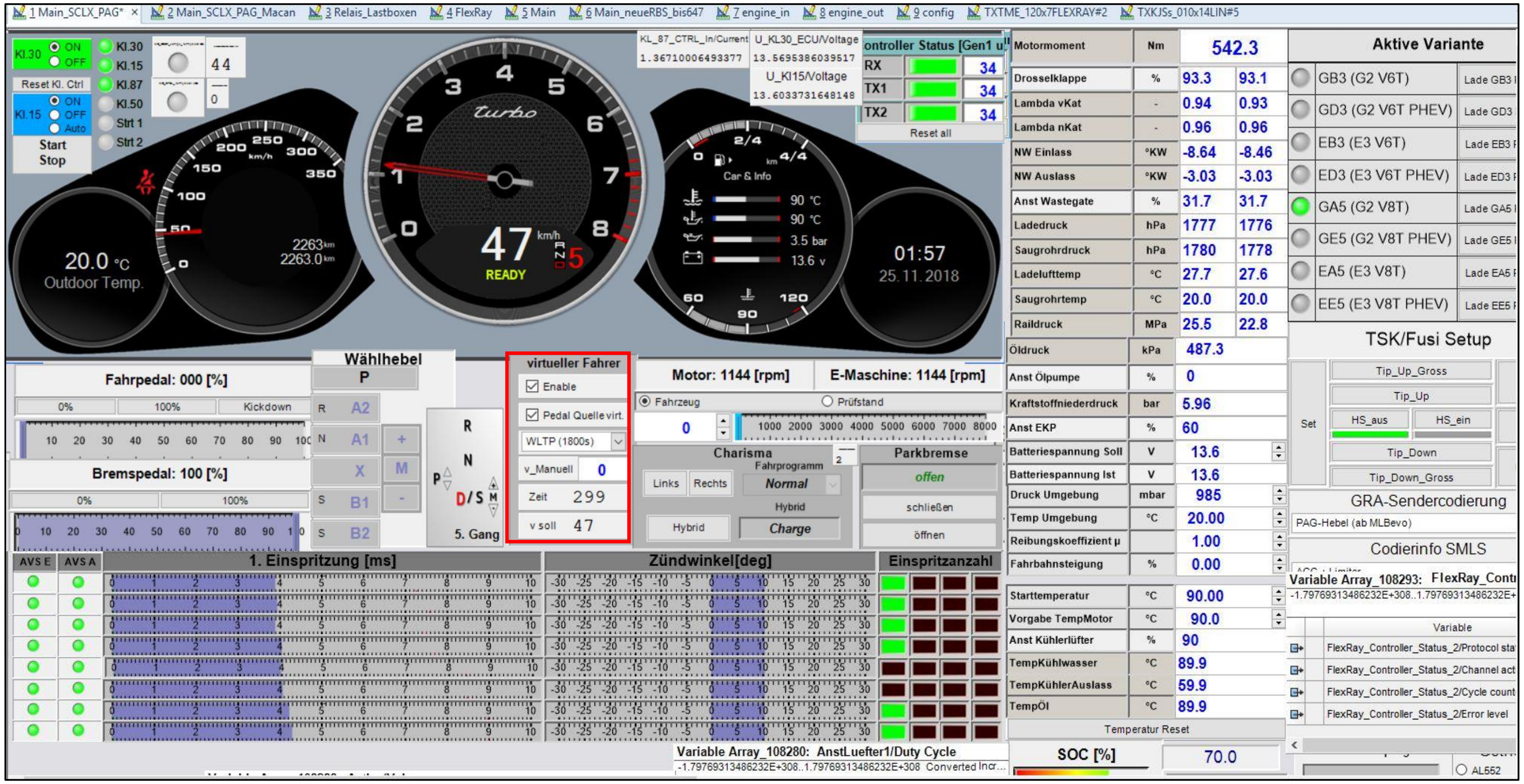
Table 4: Descriptive parameters of the WLTP driving cycle

6.1 Implementation of WLTP cycle

Implementation of the WLTP driving cycle on the dSpace HIL to study about fuel consumption and CO_2 emissions, is done via two methods (manual and automation method) which are discussed in following section.

6.1.2 Manual Implementation

Manual implementation of WLTP driving cycle is done by controlling the driver model through HIL ControlDesk manually marked as red area in figure 33, by activating acceleration pedal, brake pedal and gear shifter. This method is only useful to understand the behavior of the vehicle performance as by using this method other tests cannot be performed in parallel consisting WLTP cycle.

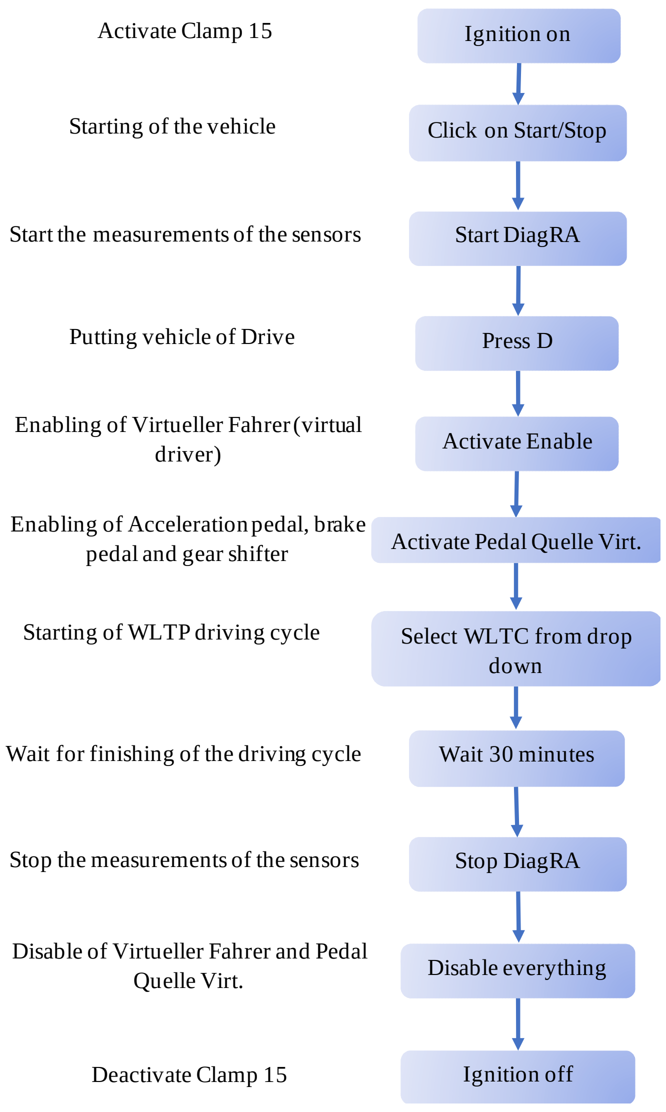


virtueller Fahrer
- [✓] Enable
- [✓] Pedal Quelle virt.
- WLTP (1800s)
- v_Manuell: 0
- Zeit: 299
- v soll: 47

Motormoment	Nm	542.3	
Drosselklappe	%	93.3	93.1
Lambda vKat	-	0.94	0.93
Lambda nKat	-	0.96	0.96
NW Einlass	°KW	-8.64	-8.46
NW Auslass	°KW	-3.03	-3.03
Anst Wastegate	%	31.7	31.7
Ladedruck	hPa	1777	1776
Saugrohrdruck	hPa	1780	1778
Ladelufttemp	°C	27.7	27.6
Saugrohrtemp	°C	20.0	20.0
Raildruck	MPa	25.5	22.8
Öldruck	kPa	487.3	
Anst Ölpumpe	%	0	
Kraftstoffniederdruck	bar	5.96	
Anst EKP	%	60	
Batteriespannung Soll	V	13.6	
Batteriespannung Ist	V	13.6	
Druck Umgebung	mbar	985	
Temp Umgebung	°C	20.00	
Reibungskoeffizient µ		1.00	
Fahrbahnsteigung	%	0.00	
Starttemperatur	°C	90.00	
Vorgabe TempMotor	°C	90.0	
Anst Kühlerlüfter	%	90	
TempKühlwasser	°C	89.9	
TempKühlerAuslass	°C	59.9	
TempÖl	°C	89.9	

Fig. 33: WLTP cycle manual implementation (red area)

The steps involved controlling driver model manually and implementation of WLP cycle are as follows:


Activate Clamp 15
Ignition on
Starting of the vehicle
Click on Start/Stop
Start the measurements of the sensors
Start DiagRA
Putting vehicle of Drive
Press D
Enabling of Virtueller Fahrer (virtual driver)
Activate Enable
Enabling of Acceleration pedal, brake pedal and gear shifter
Activate Pedal Quelle Virt.
Starting of WLTP driving cycle
Select WLTC from drop down
Wait for finishing of the driving cycle
Wait 30 minutes
Stop the measurements of the sensors
Stop DiagRA
Disable of Virtueller Fahrer and Pedal Quelle Virt.
Disable everything
Deactivate Clamp 15
Ignition off

6.1.2 Automation Method Implementation

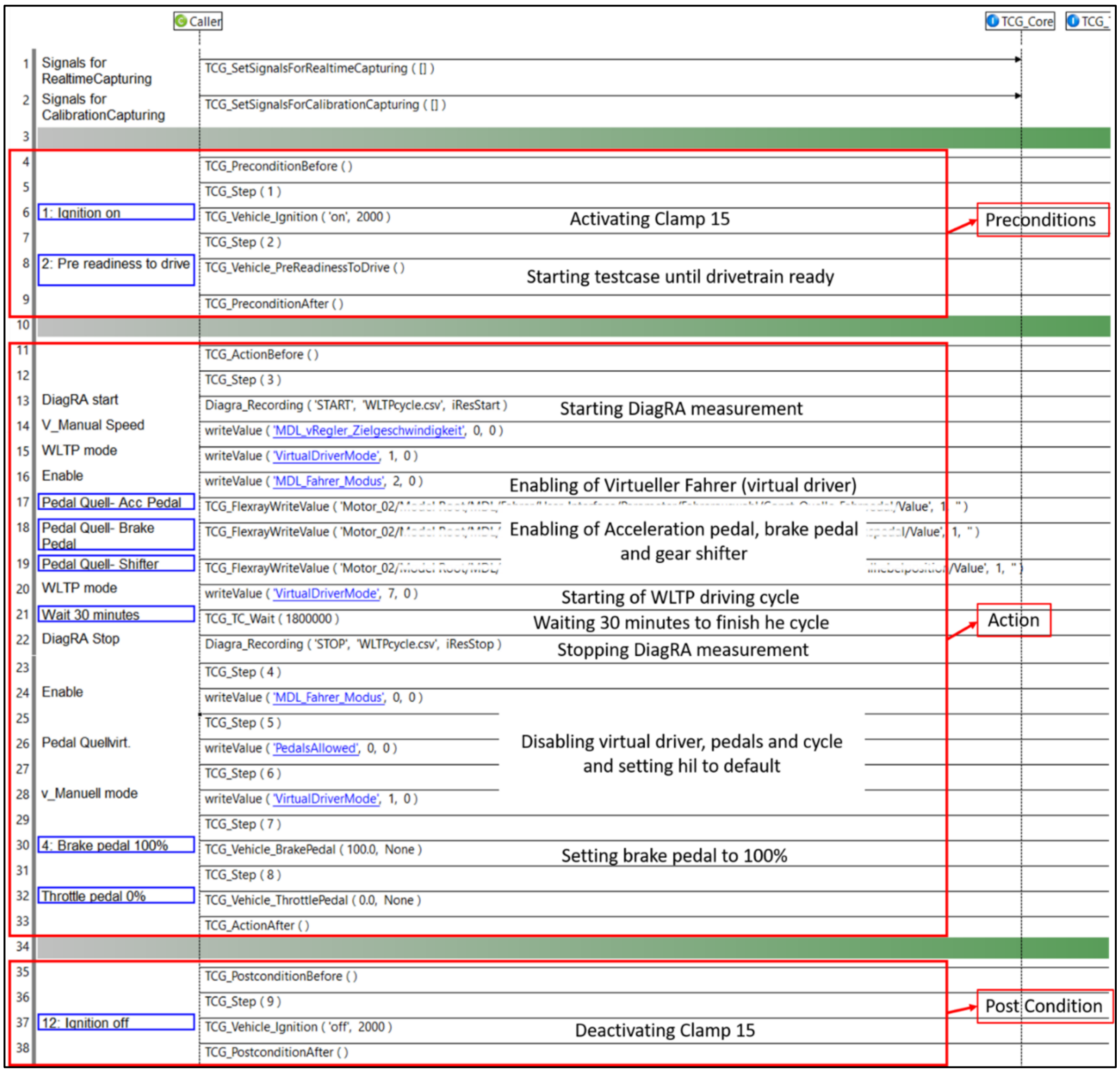
Implementation using automation method is done by preparing the TestCase in EXAM automation software which will automate all the commands which earlier has to be done manually. The test cases and the functions necessary for it are programmed manually using specific functions in EXAM starting from the requirements of DOORS.
Few important terms and definitions are discussed below to understand the EXAM automation method and executing the TestCase and preparing SequenceDiagram:

1. Package: Packages are the collection of model elements of any type. They can be used to divide the overall model into smaller, more manageable units. A package defines a namespace i.e. names of the elements contained in the package must be unique. Each model element can be referenced from the multiple package, however it belongs to only one home package.

2. TestCase: TestCase represents exactly one test flow in an abstract, formal and structured form. Each TestCase has a unique ID with a distinct test flow. Each variant of a common test flow is a separate TestCase. It can only be called by TestGroups (TestSuite).
3. SystemConfiguration: It is required for execution tests on a test system (HIL). It defines which implementation class will be used for each interface.
4. TestSuite: TestSuite defines a self-contained test topic and its content. It is organized by TestGroups referencing TestCase.
5. Variable Mapping: It is a container for mapping class instances and represents the interface to external platform variables (dspace hil). Variable mapping defines the signal path of EXAM mapping attributes in order to access the corresponding platform variable. As a result platform variables are accessible within EXAM using the get- and set- functions.
6. Test Case Generator (TCG) Syntax: TCG Syntax enables a formal description of test cases in DOORS, that can be synched to EXAM with the Synchronizer Plugin and then automatically generate the Implementation of the test case with the TCG plugin. This enables a fast and convenient workflow from specification to test execution.
7. Porsche Master Implementation (PMI): To simplify the programming work when programming test cases in EXAM, Porsche created the Porsche Master Implementation (PMI), representing 'functions' that will run before every test suite starts. It is an obligation to use this PMI whenever new testing systems are developed, especially when those systems have to use ECU diagnostics. EXAM has to be configured in a way that this PMI will run before every test suite will start.
8. SequenceDiagram: It is a Unified Modeling Language (ULM) diagram and it is the most important diagrams in EXAM. They model a series of messages (operation calls) between objects over a specific period.

   To prepare SequenceDiagram various TCG commands and shortname functions are used, which are prepared by the Porsche Engineering Services engineers in past. These commands simplify and help engineers to prepare a TestCases quickly

EXAM gives the possibility to create functions in Python and drag-and-drop them into the right order to create a specific test cases. After the execution of the TestCase, coupled into a TestSuite, EXAM generates reports. These reports are then studied over and relevant decision is made whether or not to run the failed test case again, manually or even automated and starts to search for the causes of the failure of the test case. To implement the WLTP cycle, the driver model of HIL is controlled via acceleration pedal, brake pedal and gear shifting. To automate this process, steps are prepared in the SequenceDiagram using TCG functions (in blue box) and TCG FlexRay writevalue paths as shown in figure 34.



Caller | TCG_Core | TCG_

1	Signals for RealtimeCapturing	TCG_SetSignalsForRealtimeCapturing ([])	
2	Signals for CalibrationCapturing	TCG_SetSignalsForCalibrationCapturing ([])	
3			
4		TCG_PreconditionBefore ( )	
5		TCG_Step (1)	
6	1: Ignition on	TCG_Vehicle_Ignition ('on', 2000)	Activating Clamp 15
7		TCG_Step (2)	
8	2: Pre readiness to drive	TCG_Vehicle_PreReadinessToDrive ( )	Starting testcase until drivetrain ready
9		TCG_PreconditionAfter ( )	
10			
11		TCG_ActionBefore ( )	
12		TCG_Step (3)	
13	DiagRA start	Diagra_Recording ('START', 'WLTPcycle.csv', iResStart)	Starting DiagRA measurement
14	V_Manual Speed	writeValue ('MDL_vRegler_Zielgeschwindigkeit', 0, 0)	
15	WLTP mode	writeValue ('VirtualDriverMode', 1, 0)	
16	Enable	writeValue ('MDL_Fahrer_Modus', 2, 0)	Enabling of Virtueller Fahrer (virtual driver)
17	Pedal Quell- Acc Pedal	TCG_FlexrayWriteValue ('Motor_02/[illegible]/Value', 1 [illegible] '')	Enabling of Acceleration pedal, brake pedal and gear shifter
18	Pedal Quell- Brake Pedal	TCG_FlexrayWriteValue ('Motor_02/[illegible]/Value', 1, '')	
19	Pedal Quell- Shifter	TCG_FlexrayWriteValue ('Motor_02/[illegible]/Value', 1, '')	
20	WLTP mode	writeValue ('VirtualDriverMode', 7, 0)	Starting of WLTP driving cycle
21	Wait 30 minutes	TCG_TC_Wait (1800000)	Waiting 30 minutes to finish he cycle
22	DiagRA Stop	Diagra_Recording ('STOP', 'WLTPcycle.csv', iResStop)	Stopping DiagRA measurement
23		TCG_Step (4)	
24	Enable	writeValue ('MDL_Fahrer_Modus', 0, 0)	
25		TCG_Step (5)	Disabling virtual driver, pedals and cycle and setting hil to default
26	Pedal Quellvirt.	writeValue ('PedalsAllowed', 0, 0)	
27		TCG_Step (6)	
28	v_Manuell mode	writeValue ('VirtualDriverMode', 1, 0)	
29		TCG_Step (7)	
30	4: Brake pedal 100%	TCG_Vehicle_BrakePedal (100.0, None)	Setting brake pedal to 100%
31		TCG_Step (8)	
32	Throttle pedal 0%	TCG_Vehicle_ThrottlePedal (0.0, None)	
33		TCG_ActionAfter ( )	
34			
35		TCG_PostconditionBefore ( )	
36		TCG_Step (9)	
37	12: Ignition off	TCG_Vehicle_Ignition ('off', 2000)	Deactivating Clamp 15
38		TCG_PostconditionAfter ( )	

Fig. 34: WLTP cycle sequence diagram

6.2 HIL Virtual Driver Behavior

The vehicle driver system is shown in Figure 35. The driver controls the vehicle using steering, acceleration, and braking as an inputs. The vehicle responds, according to the information driver provided in terms of path to be followed, orientation (i.e. yaw angle), lead time and vehicle in front distance.

Vibrations and vehicle acoustic variations are feedback values used by the driver to assess the current and future vehicle status, with respect to the intended response. In addition, the driver receives information through the control devices. For e.g. in a low road friction condition, driver will reduce the torque feedback on the steering wheel and activation of ABS is noticed through vibrations in the brake pedal. Further, the vehicle will face external disturbances such as aerodynamic forces, road irregularities, and road friction differences.
Consequently, the vehicle will not respond in the same way to the driver want it to be according the inputs. This is due to the impact of external disturbances and the limited ability of the driver to control the vehicle accurately means that the driver is constantly correcting his input to the vehicle. The driver responds to the vehicle behavior to fulfill a certain task, which is closed-loop behavior, in contrast to the situation where one is considering vehicle response to driver input without driver feedback is open-loop behavior. In dSpace HIL system, close loop behavior model is used.

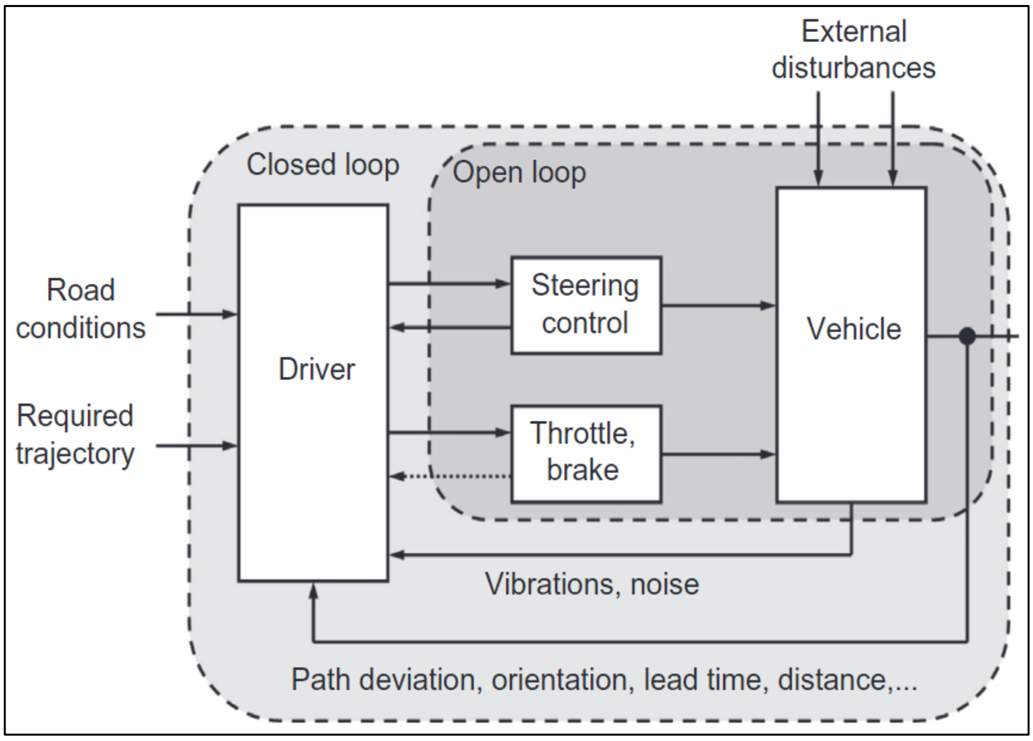


Fig. 35: Vehicle driver system [30]

Above is the overview of how the driver model behavior is simulated in the HIL System. As shown in the figure 36, where the vehicle behavior is studied on the basis of the WLTP cycle. The behavior of the driver is quite prominent, HIL virtual driver almost completely followed the inputs of the cycle.

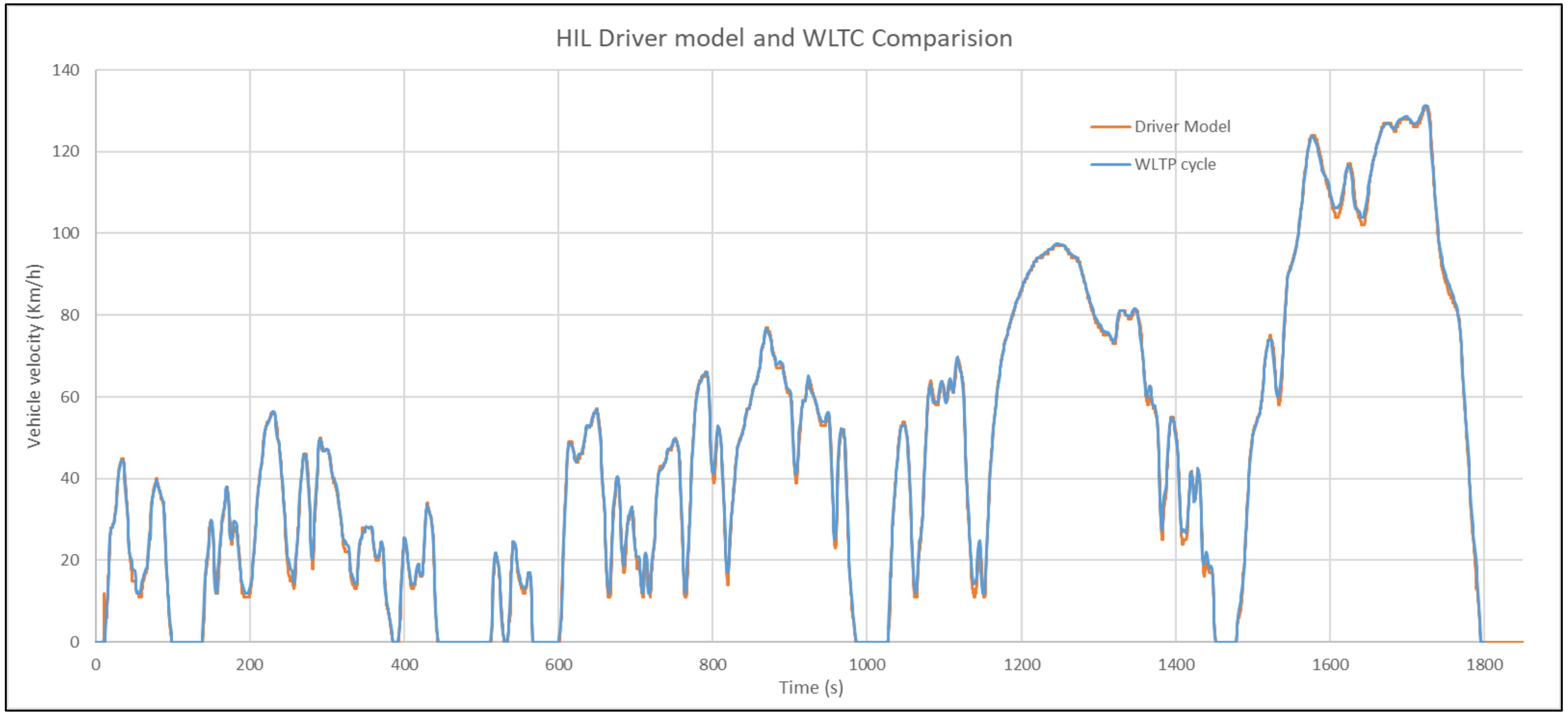


Fig. 36: HIL Virtual driver behavior on WLTP Cycle

Chapter 7
Calculations and Assumptions

In this chapter, the calculations and assumptions are discussed which are required to calculate the fuel consumptions and CO_2 emissions on WLTP cycle. The cycle is run on the dSpace KoVoMo HIL system control desk via the EXAM TestCase. The sensors data are extracted from DiagRA Software which are used for the calculations.

7.1 Fuel Consumption

The accurate calculation of following two parameters are very important to estimate fuel efficiency and vehicle emissions.

1. The amount of <u>air</u> entering in the engine combustion chamber i.e. Mass Air Flow (MAF) discussed in section 7.1.1
2. The amount of <u>fuel</u> entering in the engine combustion chamber i.e. Mass Fuel Flow (MFF) discussed in section 7.1.2

In urban vehicular pollution monitoring tool, the speed density method is most adopted technique to determine Mass Air Flow (MAF). In this work as well, this method has been implemented to calculate Mass Air Flow (MAF) which will be used to further calculate Mass Fuel Flow (MFF).
It is important to note that, to carry out perfect combustion, ECU control several components which perform the task of defining the amount of air required. In addition, ECU calculate the ideal air temperature and pressure, the current value of the vehicle and the actual air flow that the engine needs at any given time.

7.1.1 Mass Air Flow (MAF)

Speed density method is used to measure mass air flow, it performs an estimation based on the ideal gas law. To do this, it uses readings of intake manifold temperature and air pressure sensors in the vehicle, in addition to using the volumetric efficiency of the engine.
This method can be used with the manifold absolute pressure sensor, as it measures the absolute pressure. This way of predicting Mass air flow is done when MAF sensor is not available in the vehicle (which is the case in this thesis)

Starting with the law of physics, we observe a series of relations between the temperature, pressure, and volume of the gas, Equation (1) is the mathematical representation of this law

$$P.V = n.R.T \tag{1}$$

The terms of above Equation are explained as follows:

- P is the pressure in the combustion chamber and can be determined by means of the MAP (Manifold Absolute Pressure) sensor in kPa
- V is the volume of the combustion chambers in the engine cylinders and can be measured in units of volume as Liters (L) or cubic centimeters (cm^3)
- R is the ideal gas constant. Its value is approximately 8.3145 J/mol.K
- T is the gas temperature. It can be acquired by the IAT (Intake Absolute Temperature) sensor in K.
- n is the number of moles.

By rearranging the terms of equation (1) the amount of moles (n) of the given amount of air flowing through the air intake before the combustion can be calculated.
Using number of moles, the mass of air (m_{air}) can be calculated by multiplying it by the molecular weight (molar mass, represented by M_{air}) of the air, as described in Equation (2).

$$m_{air} = n.M_{air} \tag{2}$$

Using the equation (1) and (2) air mass can be obtained in the form of equation (3)

$$m_{air} = \frac{P.V}{R.T}.M_{air} \tag{3}$$

Equation (3) is only valid if the engine has a Volumetric Efficiency (VE) of 100%. VE is the ratio between the air-fuel mixture volume that each cylinder admits and the nominal cylinder capacity, The VE can be found using the following relation (4):

$$VE = \frac{V_{intake}}{V_{nominal}} \times 100\% \tag{4}$$

Where, V_{intake} represents the real volume of intake air supported by the cylinders and $V_{nominal}$ is the theoretical volume of the engine. According to the [31] the turbocharged V8 engine range of VE varies from 80% to 92% so for the calculation of MAF on a real vehicle it's estimated using HIL MAF volumetric efficiency.

Thus, using Equation (3 and 4) coupled with the engine Revolutions per Minute (RPM), we can finally obtain the mass flow through the intake at given amount of time. The RPM the engine is working can be obtained via OBD-II or via CAN-FD using DiagRA.

During the vehicle operation, in a four-stroke engine, there are two steps of air intake, i.e. in a complete cycle, air flows inside the engine two times. With that information, we can calculate the number of times air enters the engine per second by dividing the RPM by 2×60. Incorporating this into above equations gives the mass air flow ($\dot{m}_{air}$), shown by (5).

$$\dot{m}_{air}(kg/h) = \frac{P.V}{R.T}.M_{air}.VE.\ \frac{RPM}{120} \tag{5}$$

The value obtained of turbocharged engine by this equation corresponds to an equivalent to that obtained directly by the MAF sensor without EGR (Exhaust gas recirculation) as shown in figure 37. As explained in [36] the overall results suggest that the unpredictability of the generic speed-density method is in the order of 10% throughout most of the engine operating range, but increasing to tens of percent where high-volume exhaust gas recirculation (EGR) is used.

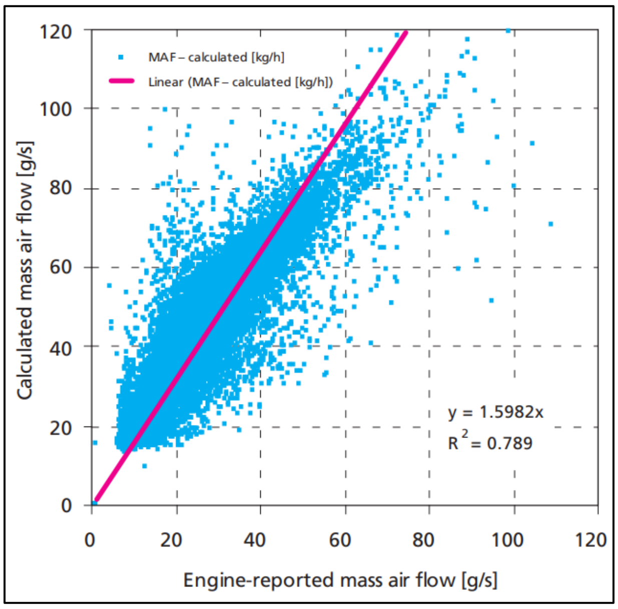


Fig. 37: Comparison of calculated mass air flow with mass air flow reported by the ECU [36]

For validation of results (discussed in chapter 7), the MAF calculation is based on speed density method due to the unavailability of the MAF sensor values from OBD II in Porsche Panamera 4. On the other hand, while calculating fuel consumption on the WLTP cycle, the MAF values are used from CAN communication using DiagRA. Few of the assumptions are discussed as followed:

- The assumption of ignoring EGR can affect the final results as all the Porsche vehicles are equipped with EGR system. EGR is mainly active during partial engine loads and at low and medium engine speed areas, where oxygen is in excess. In the high engine load (torque), the EGR system is disabled, the cylinders being filled only with air, ready for combustion [37]. According to [38] the research results indicate improvement of fuel consumption improvement with cooled EGR under conditions where stoichiometric operation cannot be maintained.

- Effect of Volumetric Efficiency: Volumetric efficiency in a combustion engine depends on many factors like Fuel type, Air/Fuel ratio, fuel heat of vaporization, Intake manifold temperature and pressure, Engine speed, Compression ratio, port design intake and exhaust, etc. [34]. So, the assumption in calculating the MAF using a generic value can affect the final results as the volumetric efficiency is not fixed and it varies according to the above mentioned factors.

7.1.2 Mass Fuel Flow (MFF)

Mass fuel flow $\dot{m}_{fuel}(kg/h)$ depends on actual air fuel ratio (A/F) and Mass air flow $\dot{m}_{air}(kg/h)$. Before going into detailed calculation, it is important to define the equivalence Air-Fuel Ratio (λ) and understand how this relates to $\dot{m}_{fuel}(kg/h)$ and $\dot{m}_{air}(kg/h)$.

For a certain amount of fuel, there is a perfect amount of oxygen required for the combustion to be ideal. This factor is called the Stoichiometric Air-Fuel Ratio (A/F_{Stioc}). For the perfect combustion of petrol without any reactants, the proportion considered to be an ideal mix of fuel and oxygen has A/F_{Stioc} of 14.7:1. That is, it takes 14.7 kilogram of air for every 1 kilogram of fuel entering the combustion chamber. In alcohol vehicles, this ratio is 9:1, while in diesel vehicles, it is 14.6:1.

The equivalence Air-Fuel Ratio (λ) can be mathematically described by Equation (6), being only a division between the actual air/fuel ratio (A/F) and stoichiometric air/fuel ratio (A/F_{stoic}). This value is controlled and monitored by the oxygen sensors attached before and after (for few vehicle models) catalytic convertor.

$$\lambda = \frac{A/F}{A/F_{Stoic}} \tag{6}$$

To calculate A/F from a given λ, the measured λ is multiplied by the stoichiometric AFR (A/F_{stoic}) for that fuel. The actual air/fuel ratio (A/F) is calculated as equation (7).

$$A/F = \lambda \times A/F_{Stoic} = \frac{\dot{m}_{air}(kg/h)}{\dot{m}_{fuel}(kg/h)} \tag{7}$$

The OBD-II standard establishes codes to acquire the mass air flow directly, or, in case the vehicle does not have an MAF sensor, it also provide with the access to other sensors that can be used to obtain such a value (speed density method). However, it does not provide a method to directly obtain the amount of fuel injected in the combustion chamber, which is ideal for this study. Considering that the λ is a known value and mass air flow as calculated in section 7.1.1, the mass of fuel can be found by reordering Equation (7) as presented in Equation (8).

$$\dot{m}_{fuel}(kg/h) = \frac{\dot{m}_{air}(kg/h)}{A/F} \tag{8}$$

The fuel volume flow rate (represented in units of volume over time) to find the relation between the volume of fuel consumed. To get this ratio of the fuel mass flow rate and the respective fuel density is calculated as Equation (9).

$$\dot{V}_{fuel}(l/h) = \frac{\dot{m}_{fuel}(kg/h)}{\rho_{fuel}(kg/l)} \tag{9}$$

Petrol fuel is used for the study, and in Czech Republic its density is in the range 0.73 Kg/l - 0.78 Kg/l, so ρ_{fuel} = 0.75 Kg/l is assumed [32]

Now, using the equation (8) and multiplying it vehicle velocity obtained from sensor, fuel consumption is calculated as in Equation (10)

$$Fuel\ Consumption\ (l/100\ km) = \frac{\dot{V}_{fuel}(l/h)\,.100}{V_{velocity}(km/h)} \tag{10}$$

7.2 Emissions (CO_2)

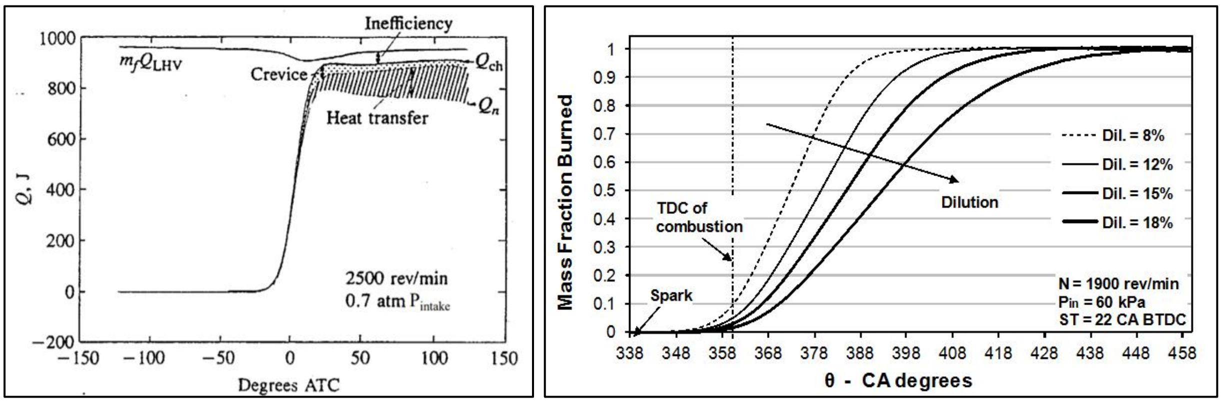
As CO_2 is a greenhouse gas and plays important role in global warming, the scope of this study will be limited to study of CO_2 in the emissions. To estimate the amount of CO_2 in this work, complete combustion [33] is assumed without any pollutants e.g. UHC, CO, PM, NOx, etc. It is important to note that, complete oxidation of simple hydrocarbon fuels (C_nH_m) forms carbon dioxide (CO_2) from all of the carbon and water (H_2O) from the hydrogen.
The mass of emitted CO_2 is closely linked to the volume of fuel burned stated in equation (8). This relationship is expressed in the Equation (10), where the fuel volume flow rate is multiplied by the mass of CO_2 generated by the combustion of one liter of fuel. The outcome of this equation is the estimate of the CO_2 mass flow rate released per second.

$$\dot{m}_{CO_2}\ (g/h) = \dot{V}_{fuel}(l/h)\,.CO_{2\,per\,litre} \tag{10}$$

Where, $CO_{2\ per\ litre}$ is 2392 g/l, and is calculated as followed:
1 liter of petrol weighs 750 grams (ρ_{fuel} = 0.75 Kg/l). Petrol contains for 87% of carbon or 652 grams of carbon per liter of petrol. In order to convert this carbon to CO_2 1740 grams of oxygen is needed. The sum is then 652 + 1740 = 2392 grams of CO_2/liter of petrol.

Therefore,

$$\dot{m}_{CO_2}\ (g/km) = \frac{Fuel\ consumption\ (l/100km)\,.CO_{2\,per\,litre}}{100} \tag{11}$$



(a) (b)

Fig. 38: (a) Results of heat-release analysis showing the combustion inefficiency and the corrections due to heat transfer and Crevice effect [34]. (b) Mass fraction dependency on one of the many factors on the amount of dilution [35].

This value gives an estimate in the calculation of the CO_2 emissions and few of the assumptions are discussed as followed:

- The Mass Fraction Burn (MFB) and Heat Release Rate (HRR) shows the amount of fuel burned and the rate of burning throughout the combustion process in an internal combustion engine. These value are predicted using two zone complex models, to make calculation simple, combustion efficiency and mass fraction burned is assumed to be hundred percent. But in reality it is not the case as shown in figure 38.
- Effect of dissociation: At very high combustion temperatures usually around 1700K [34], the CO_2 formed during combustion, starts to break (dissociate) into CO and O_2. This dissociation is an endothermic reaction and it absorbs heat from its environment, resulting in a loss in the net energy created and therefore, reducing the efficiency of the IC engine. This effect is also neglected for this study.

$$CO_2 \rightleftarrows CO + O_2 \qquad (at\ high\ temperature)$$

Chapter 8
Results and Discussion

This chapter is mainly divided into two sections, first section (section 8.1.) involves the validation of the dSpace Hardware in loop (HIL) at different driving cycle phases measured on the real vehicle (Porsche Panamera 4) with primary focus on fuel consumption calculations.
The second section (section 8.2.) utilizes the validation results from first section and applies them to calculate the WLTP cycle on HIL discussing in detail the fuel consumption and CO_2 emissions. At the end, in section 8.3 the dSpace HIL Fault Code Diagnostics is discussed in detail.

8.1 HIL Validation

To validate the dSpace HIL a simple driving cycle is performed on a real vehicle (Porsche Panamera 4).
For Real data collection from Porsche Panamera 4, the OBD-II communication interface is used. Few of the codes values of Mode 1 PIDs are used to require data from the ECU as shown in table 5.
For Hardware-In-Loop (HIL) ECU data collection, CAN-FD communication protocol and DiagRA software are used.

PID	**PARAMETER**
0 X 05	Engine Coolant Temperature
0 X 06	Short Term Fuel trim
0 X 34	Lambda
0 X 87	Intake Manifold Absolute Pressure
0 X 77	Intake Absolute Temperature
0 X 0C	Engine RPM
0 X 0D	Vehicle Speed

Table 5: PID and Parameters selected for measurement in real vehicle

It is important to note that the only difference between both Real and HIL data extraction values is absence of Mass Air Flow sensor (MAF) real vehicle (Porsche Panamera 4). As discussed in section 7.1.1 the real vehicle MAF values were calculated using speed density method using HIL volumetric efficiency values at different phases of the driving cycle.

8.1.1 Approach for HIL Validation

The approach which is used to validate HIL is by Dependency Method. As shown in the dependency diagram (figure 39), the fuel consumption is dependent on various factors which subsequently dictate other parameters. This approach will help to move close to pointing the problem in the performance of the HIL.

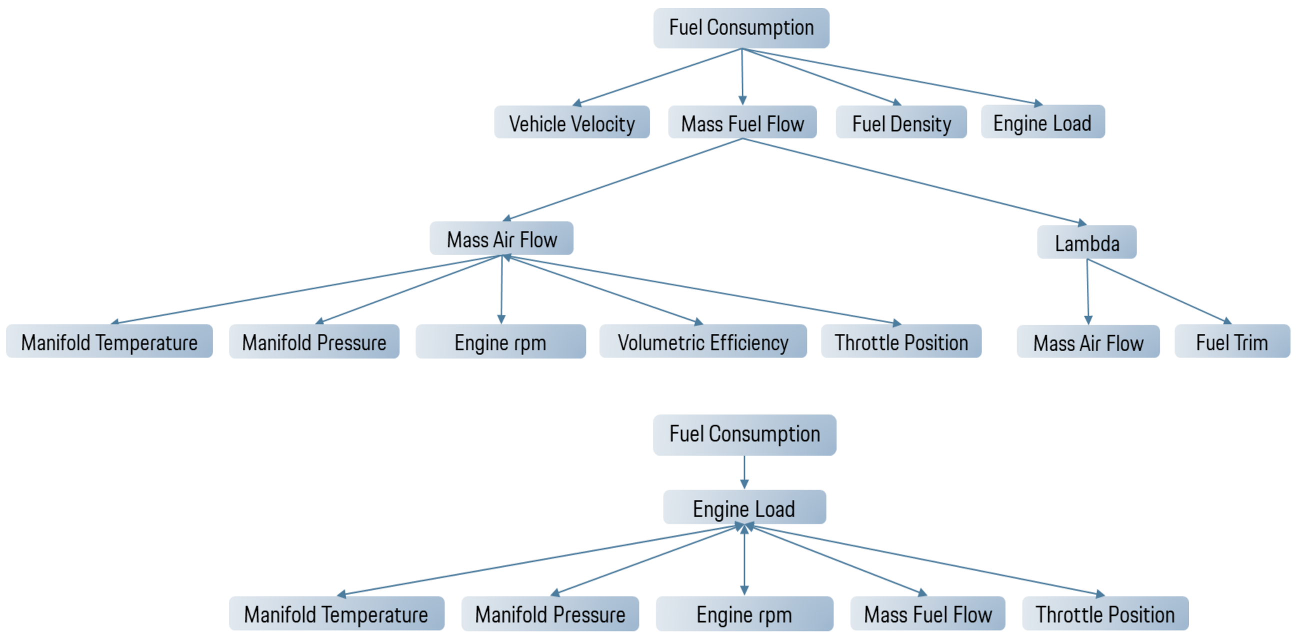


Fig. 39: Fuel consumption dependency diagram

To focus on the HIL performance the real driving cycle which is performed on the road and HIL is divided into three phases as shown in figure 40.

1. Acceleration Phase: In this phase the vehicle is accelerated using full throttle to reach the desired speed. As the real vehicle is already driving until 10 seconds with varying throttle so first ten seconds are ignored. This phase is only studied when the throttle of both HIL and real vehicle are at maximum value i.e. after 10 sec.

2. Constant driving Phase: In this the vehicle is driven in the constant velocity for few seconds.

3. De-acceleration Phase: This phase is same as engine braking, with no acceleration and no braking the vehicle is deaccelerated using the engine brake.

Further, using these results and observation from HIL validation in different phases, a good conclusion can be achieved about the behavior of HIL driving on WLTP cycle.

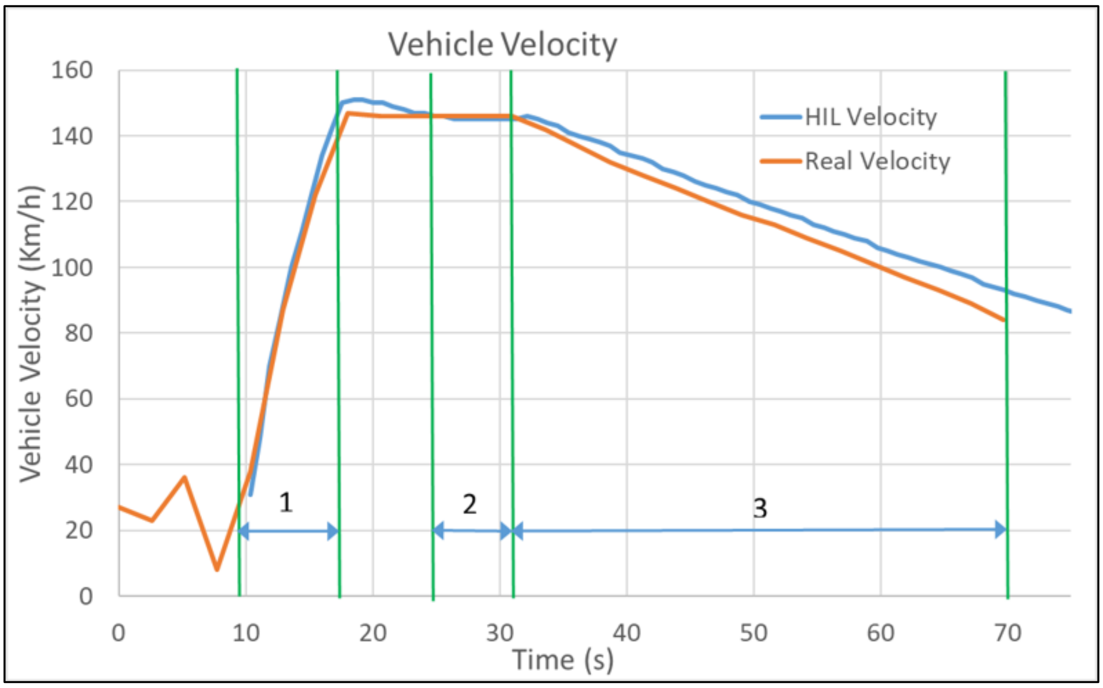


Fig. 40: Vehicle Velocity phases 1. Acceleration 2. Constant driving 3. De-acceleration

Before moving forward with the results and observation of every phases, as mentioned above the calculation of the Mass Air Flow is done using the average values of HIL Volumetric Efficiency i.e. 87%, 67% and 67 % for phase 1, 2 and 3 resp. as shown in figure 41. As it's observed in the acceleration phase the efficiency is going above hundred percent this is typical values of Turbocharged V6 engine.

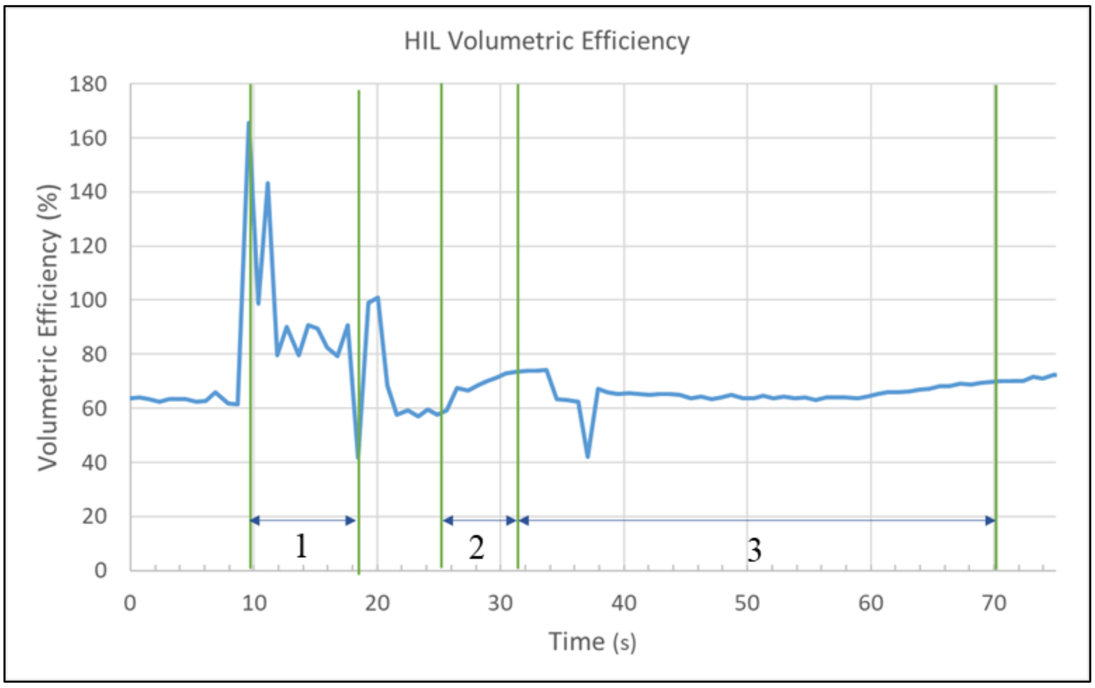


Fig. 41: Volumetric efficiency at different phases at HIL driving cycle

8.1.2 Implementation of driving cycle on HIL

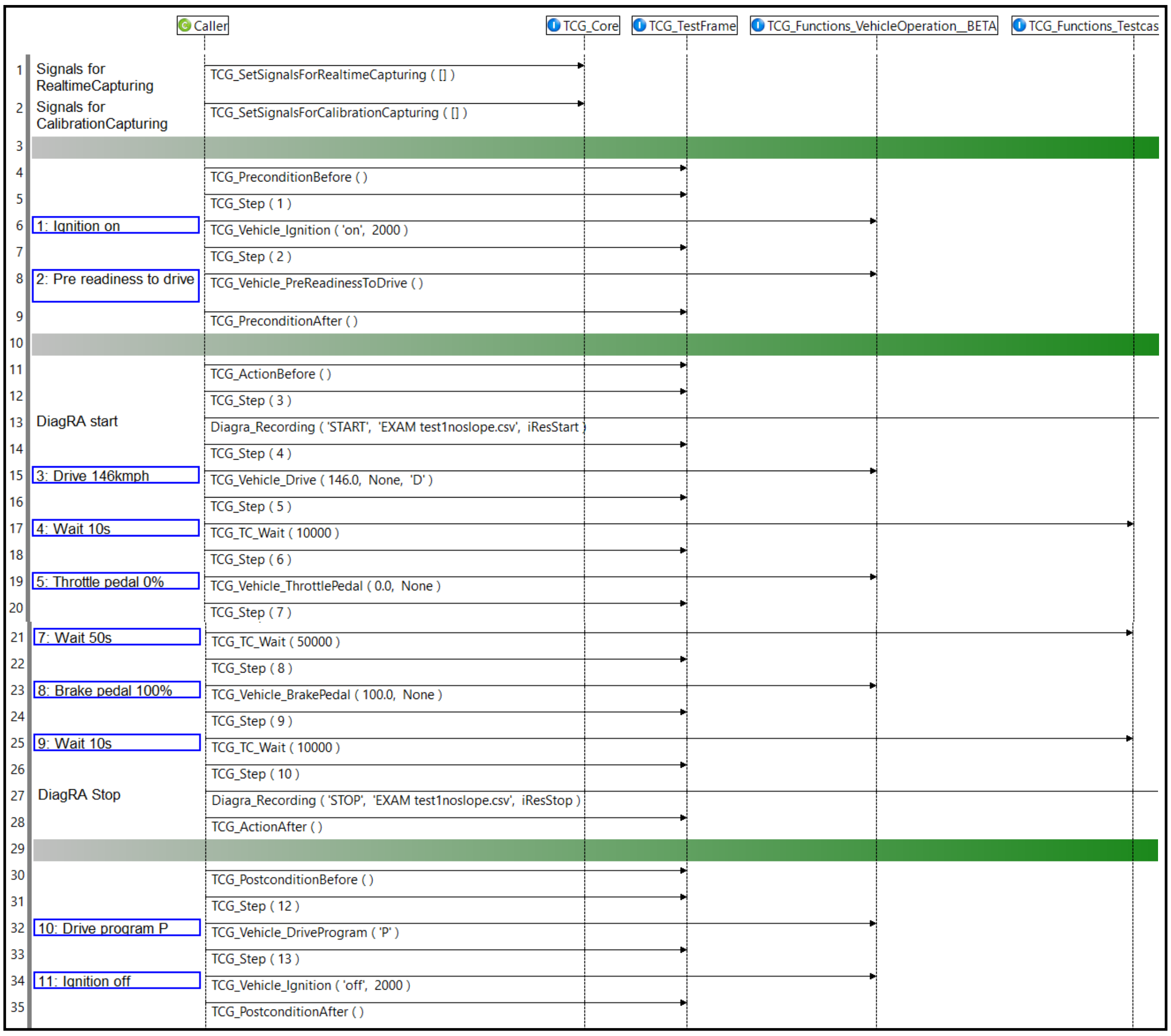
To understand the behavior of HIL, the HIL driving cycle should be very similar to the real driving cycle and it is implemented using TCG functions using EXAM automation software. The TestCase (figure 42) and the TestSequence Diagram (figure 43) of the cycle is as follows:

Precondition:
1: Ignition on
2: Pre readiness to drive

Action:
3: Drive 146kmph
4. Wait 10s
5: Throttle pedal 0%
6. Wait 50s
7. Brake pedal 100%
8. Wait 10s

Postcondition:
9: Drive program P
10: Ignition off

Fig. 42: Real driving Cycle TestCase



1	Signals for RealtimeCapturing	TCG_SetSignalsForRealtimeCapturing ([])
2	Signals for CalibrationCapturing	TCG_SetSignalsForCalibrationCapturing ([])
3		
4		TCG_PreconditionBefore ( )
5		TCG_Step (1)
6	1: Ignition on	TCG_Vehicle_Ignition ('on', 2000)
7		TCG_Step (2)
8	2: Pre readiness to drive	TCG_Vehicle_PreReadinessToDrive ( )
9		TCG_PreconditionAfter ( )
10		
11		TCG_ActionBefore ( )
12		TCG_Step (3)
13	DiagRA start	Diagra_Recording ('START', 'EXAM test1noslope.csv', iResStart)
14		TCG_Step (4)
15	3: Drive 146kmph	TCG_Vehicle_Drive (146.0, None, 'D')
16		TCG_Step (5)
17	4: Wait 10s	TCG_TC_Wait (10000)
18		TCG_Step (6)
19	5: Throttle pedal 0%	TCG_Vehicle_ThrottlePedal (0.0, None)
20		TCG_Step (7)
21	7: Wait 50s	TCG_TC_Wait (50000)
22		TCG_Step (8)
23	8: Brake pedal 100%	TCG_Vehicle_BrakePedal (100.0, None)
24		TCG_Step (9)
25	9: Wait 10s	TCG_TC_Wait (10000)
26		TCG_Step (10)
27	DiagRA Stop	Diagra_Recording ('STOP', 'EXAM test1noslope.csv', iResStop)
28		TCG_ActionAfter ( )
29		
30		TCG_PostconditionBefore ( )
31		TCG_Step (12)
32	10: Drive program P	TCG_Vehicle_DriveProgram ('P')
33		TCG_Step (13)
34	11: Ignition off	TCG_Vehicle_Ignition ('off', 2000)
35		TCG_PostconditionAfter ( )

Fig. 43: Real driving Cycle TestSequence

8.1.3 Results of Acceleration Phase

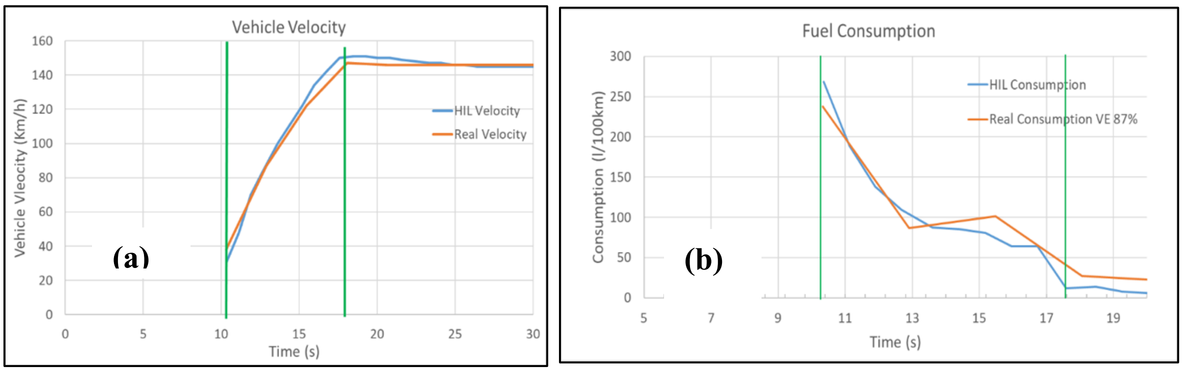


Fig. 44: HIL and Real fuel consumption at acceleration phase

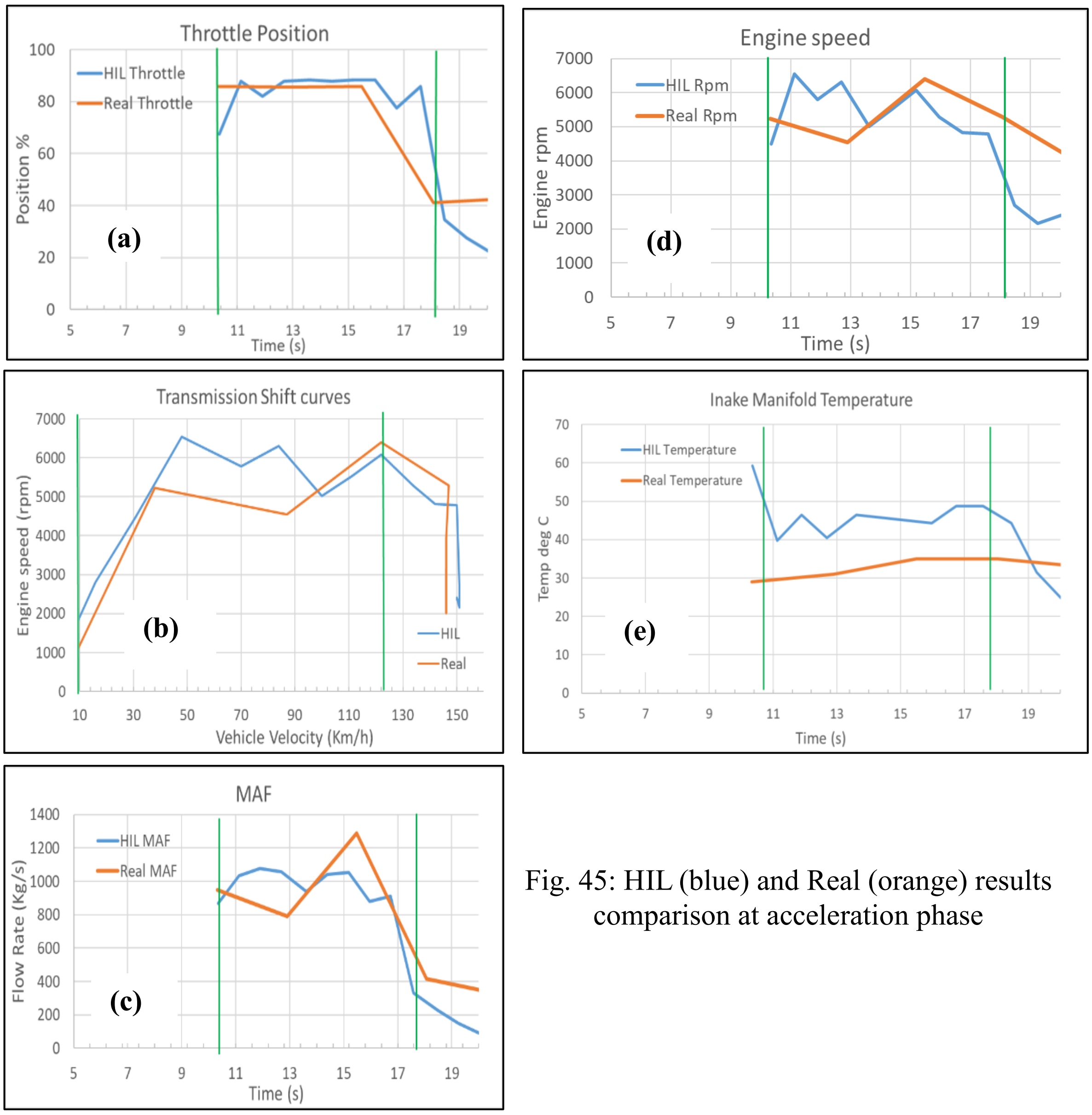


Fig. 45: HIL (blue) and Real (orange) results comparison at acceleration phase

As shown in figure 44, the fuel consumption of real vehicle and HIL results are almost following each other. The fuel consumption during acceleration phase of real vehicle is **113.0 l/100km**, while consumption calculated from HIL is **110.0 l/100km**.

According to the comparison results as shown in figure 45, few of the observations are as follows:

1. The behavior of all graphs is based on the transmission shifting curve [figure 45 (b)]. To reach 0-120 Km/h real vehicle is on 2nd gear while HIL driver model is on 3rd gear.
2. Throttle actuation [figure 45 (a)] by the real driver is gradual while at HIL it's sudden.
3. As fuel consumption is inversely proportional to the Intake Manifold temperature, the HIL is unable to maintain the constant temperature while real vehicle can. This is one of the vital reason for the overestimation of the HIL consumption.

8.1.4 Results of Constant driving Phase

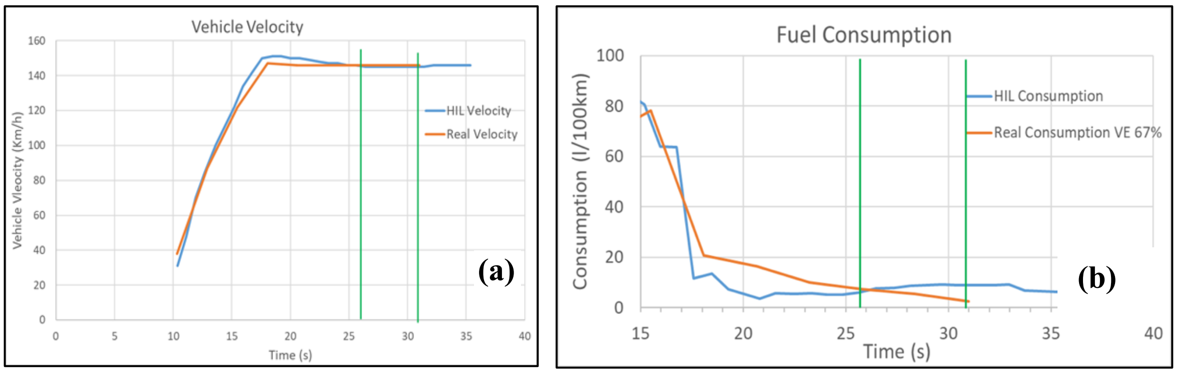
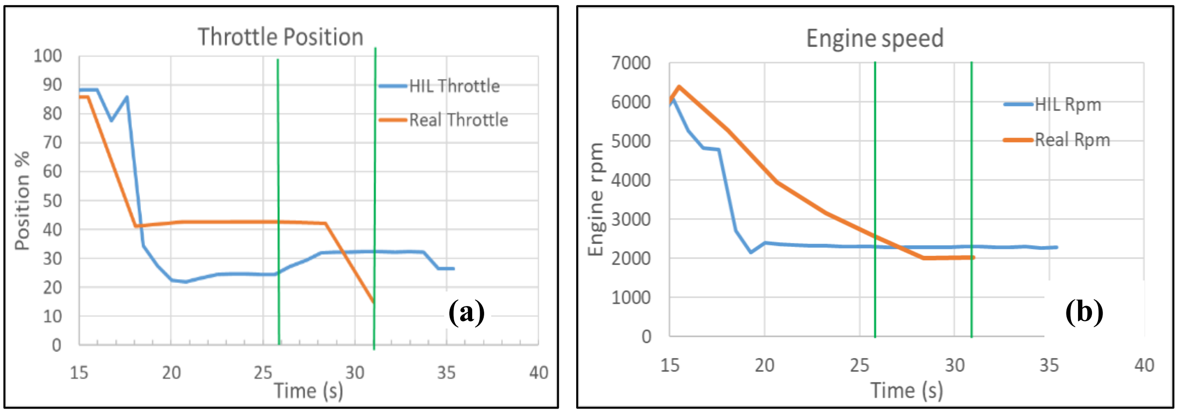


Fig. 46: HIL and Real fuel consumption at constant driving phase

As shown in figure 46, the fuel consumption of real vehicle and HIL results are almost following each other. The fuel consumption during constant speed phase of real vehicle is **5 l/100km**, while consumption calculated from HIL is **7.5 l/100km**.
In this phase, the overdrive plays a very important role to improve the fuel consumption. It is the operation in which vehicle is cruising at sustained speed with reduced engine revolutions per minute (rpm), leading to better fuel consumption.



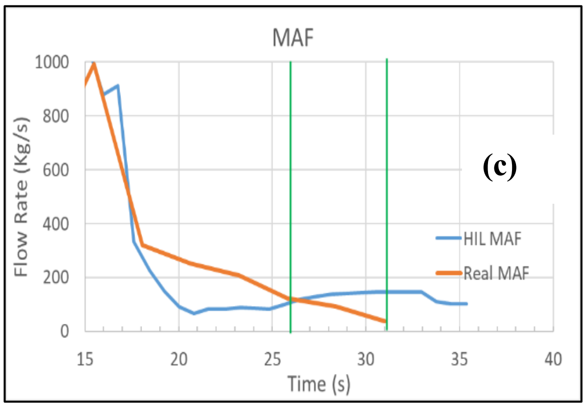


Fig. 47: HIL (blue) and Real (orange) results comparison at constant driving phase

According to the comparison results as shown in figure 47, few of the observations are as follows:

1. The fuel consumption depends upon the overdrive, which further depends upon engine speed, as observed from the [figure 47 (b)] the real vehicle rpm is less than HIL rpm.
2. Throttle actuation [figure 47 (a)] by the real driver is gradual while at HIL it's sudden. Also, throttle of HIL even at constant engine speed is increasing.

8.1.5 Results of Deceleration Phase

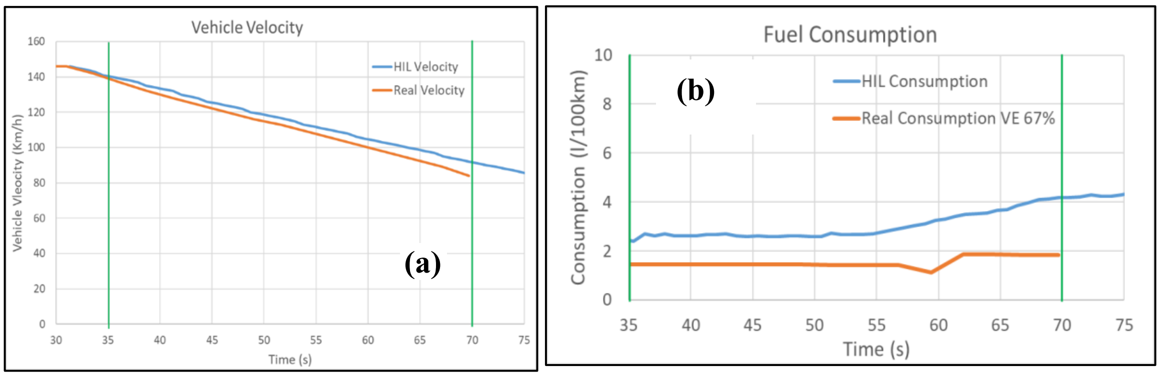


Fig. 48: HIL and Real fuel consumption at de-acceleration driving phase

As shown in figure 48, the fuel consumption of real vehicle and HIL results are almost following each other. The fuel consumption during deceleration phase of real vehicle is **1.5 l/100km**, while consumption calculated from HIL is **3.2 l/100km**. In this phase the engine-braking is focused as it plays very important part for the observation of the HIL performance. "Engine braking" refers to the braking effect occurs when the throttle valve is completely closed – which causes a strong manifold vacuum for which the cylinders have to work against. During engine braking, instead of applying footbrake the vehicle is forced to slowed down by releasing the accelerator and shifting down through gears. This causes fuel injection to cease and greatly restricting forced airflow.

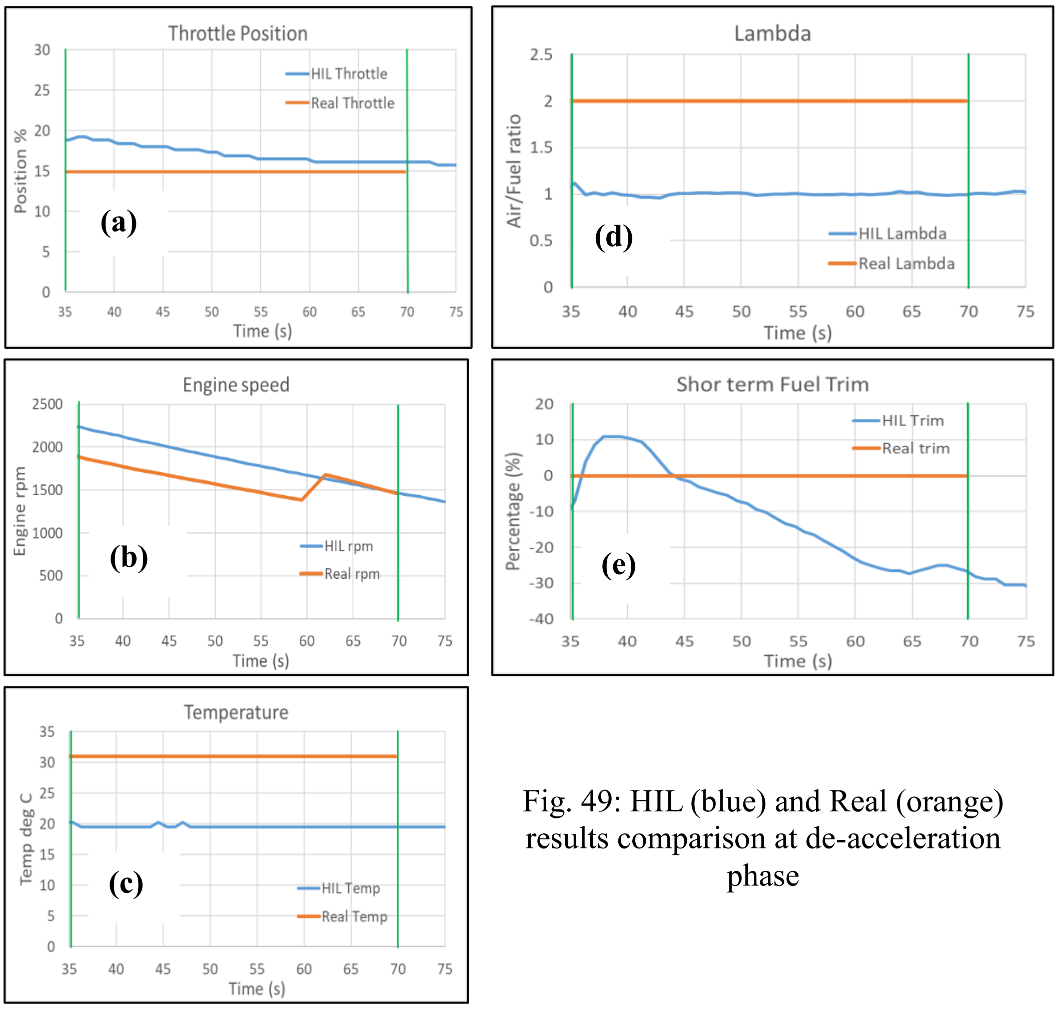


Fig. 49: HIL (blue) and Real (orange) results comparison at de-acceleration phase

According to the comparison results as shown in figure 49, few of the observations are as follows:

1. With the no accelerator pedal according to the figure 49 (a) there is still some throttle of the HIL while in real vehicle throttle is constant at 15%.
2. The short term fuel trim is varying of the HIL while real values are at zero.
3. The lambda sensor is also at stoichiometric (λ=1) for HIL and for real vehicle it's at lean region (λ>1). All these factors affect the fuel consumption.

8.1.6 Results of Full Driving Cycle

FUEL CONSUMPTION

As shown in figure 50, the fuel consumption of real vehicle and HIL results are almost following each other. The total fuel consumption during full driving cycle of real vehicle is **17.9 l/100km**, while consumption calculated from HIL is **17.0 l/100km**.

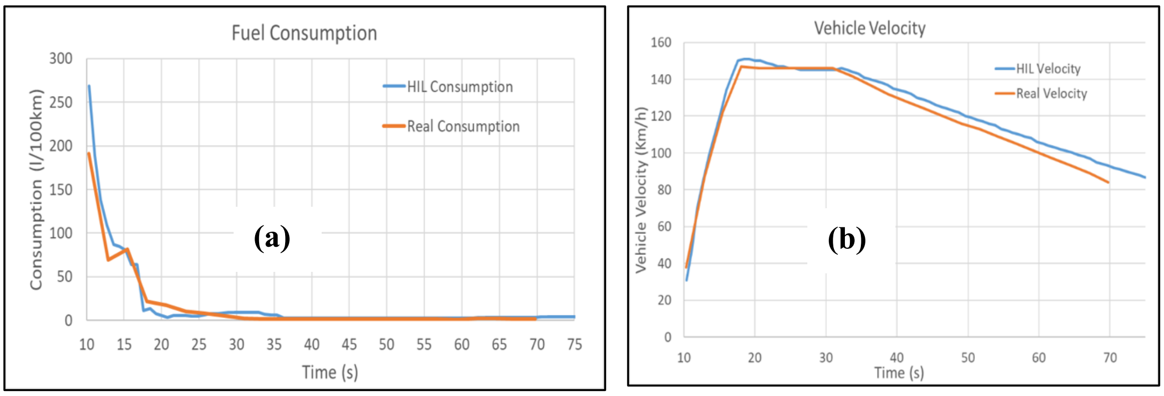


Fig. 50: HIL (blue) and Real (orange) results of fuel consumption on full driving cycle

As observed from the figure 50 (a), during accerlation phase (from 10s-19s) the fuel consumption is maximum and during idling its minimum. During the engine braking (from 35s-70s) (no braking, no throttle but vehicle in gear) the fuel injector shuts off, therefore the fuel consumption is least.
Few reasons about variation in the calculation of the fuel consumption in every phase are discussed as follows: (more reasons are explained in conclusion chapter)

Effect of lambda: The proportion of oxygen in the exhaust gases as measured by the Lambda Sensor. Lambda (λ) sensor retrieves the stoichiometric Air–fuel ratio (14.7:1 for gasoline engines with the ideal value for the combustion being 1) and this signal is sent to the ECU (Electronic Control Unit). A perfect reading is zero amps for Ideal Value. Each milliamp above zero is a part point lean and each milliamp below zero is a part point rich. According to the figure 49 (d), the lambda sensor is working perfectly for real vehicle but for the HIL it's not sending the lean values (λ=2). Lambda value two is set by default, even it's above two, and usually this happens during engine braking when fuel is cut-off.

Effect of Short term Fuel trim: The adjustment of the fuel quantity as measured by the Short-Term Fuel Trim (STFT) sensor. Depending on the signal value, in case of lean signal the ECU will increase the pulse width of the injectors to either add fuel to the air/fuel mixture (Positive fuel trim), or decrease the injector pulse width to subtract fuel from the air/fuel mixture (Negative fuel trim) in case of rich mixture and this process of adapting the injector pulse width is known as fuel trims.
As observed in figure 49(e), during acceleration and driving at constant speed the real vehicle fuel trim is adjusting the lambda (λ=1) and λ=2 during engine braking phase. But in case of Short term Fuel trim of HIL, it is unable to control fuel according to HIL Lambda values, it can be observed during engine braking phase STFT is still maintaining the lambda λ=1. This proves improper functionality of HIL as the fuel injection should stop and lambda value should go to λ=2. So HIL short term fuel trim is not performing as the real vehicle.

Effect of Engine Coolant Temperature: The temperature of the engine coolant as measured by the Engine Coolant Temperature Sensor (ECT). The PCM recognizes this signal and starts other components (such as the engine's cooling fan to maintain appropriate operating temperature). It uses different approaches which are programmed into its lookup tables for hot and cold operating conditions. This signal affects EGR (Exhaust Gas Recirculation) valve flow, PCM will not allow the (EGR) valve to open until the engine has warmed up to improve drivability [39]. If the (EGR) is allowed while the engine is still cold; it may cause a rough idle; or stalling. As shown in the figure 51, in the dSpace HIL the value is set to be constant while, as observed on the real vehicle data, the coolant temperature varies. This assumption strongly effects the EGR which in turn effects the fuel consumption.

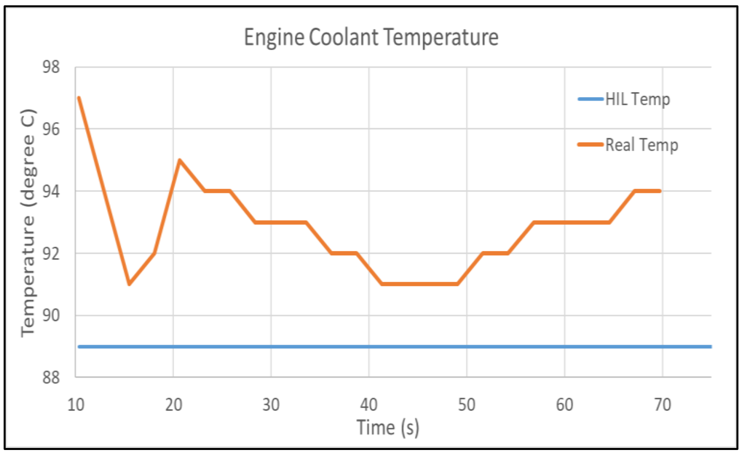


Fig. 51: Coolant temperature of HIL and real vehicle

ESTIMATED CO_2 EMISSIONS

As explained in the previous section 7.2 of WLTP CO_2 emission, the estimation is based on the assumption of complete combustion of fuel. The results obtained at different phases are as shown in table 6. The difference in CO_2 emission numbers between dSpace HIL and Real vehicle can be observed. As we have established that CO_2 emission depends on fuel consumption, the differences recorded are the result of values obtained fuel consumption at different driving phases.

Driving Phase	dSpace HIL	Real Full Driving cycle
Full driving cycle	406.6	428.16
Acceleration Phase	2631.2	2702.9
Constant driving Phase	179.4	119.6
De-acceleration Phase	76.5	35.8

Table 6: HIL and real vehicle CO_2 emission average values.

8.2 WLTP cycle results

With Porsche Panamera ECU attached to the dSpace HIL, the collection of the desired data from Engine Control Module is retrieved via DiagRA software. The sensor and ECU values which are used for estimating instantaneous fuel consumption (l/100km) and CO_2 emissions are as follows:

- Vehicle Speed
- Engine RPM
- Mass Air Flow (MAF)
- Intake Manifold Absolute Pressure (MAP)

- Intake Absolute Temperature (IAT)
- Lambda
- Short term fuel trim

8.2.1 Fuel Consumption

As showed in figure 52, the vehicle speed has effect on the fuel consumption in different phases of WLTP cycle. The fuel consumption increases when the vehicle speed increases (during acceleration), meaning that it is mainly affected by ineffective driving and traffic.

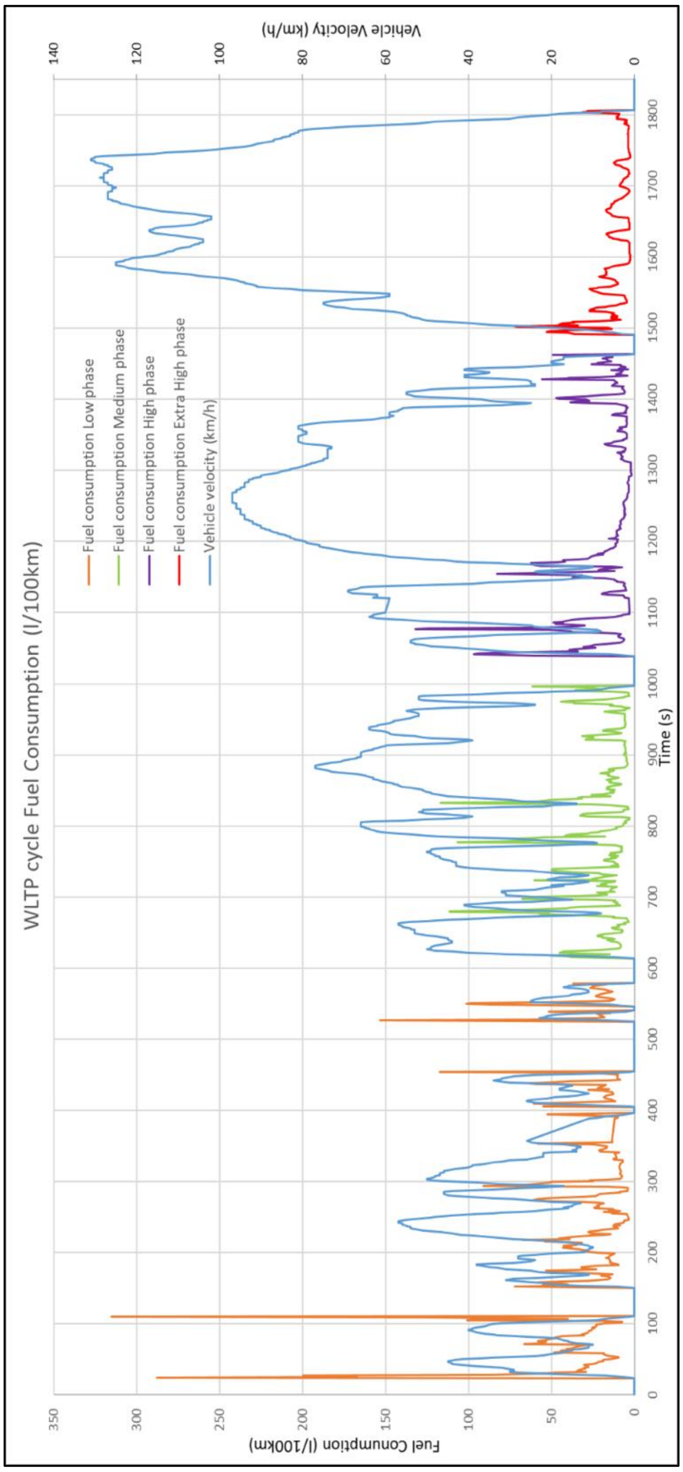


Fig. 52: Vehicle speed versus WLTP cycle fuel consumption showing Low Phase (orange), Medium Phase (Green), High Phase (Purple) and Extra High Phase (Red).

At 20s and 110s sudden acceleration and braking occurs respectively. During this time, spike in fuel consumption is recorded whereas during normal driving (without aggressive acceleration–deceleration) the fuel consumption is reduced. While, low fuel consumption has been seen when vehicle is cruising at higher speeds because vehicle tend to go into overdrive mode to save fuel. The reason of this behavior is that while cruising vehicle load is minimum, same is applicable while idling.

Figure 53 shows the cause of acceleration on the fuel consumption during WLTP cycle medium phase. According to the observation, peaks of fuel consumption are more prominent during aggressive driving.

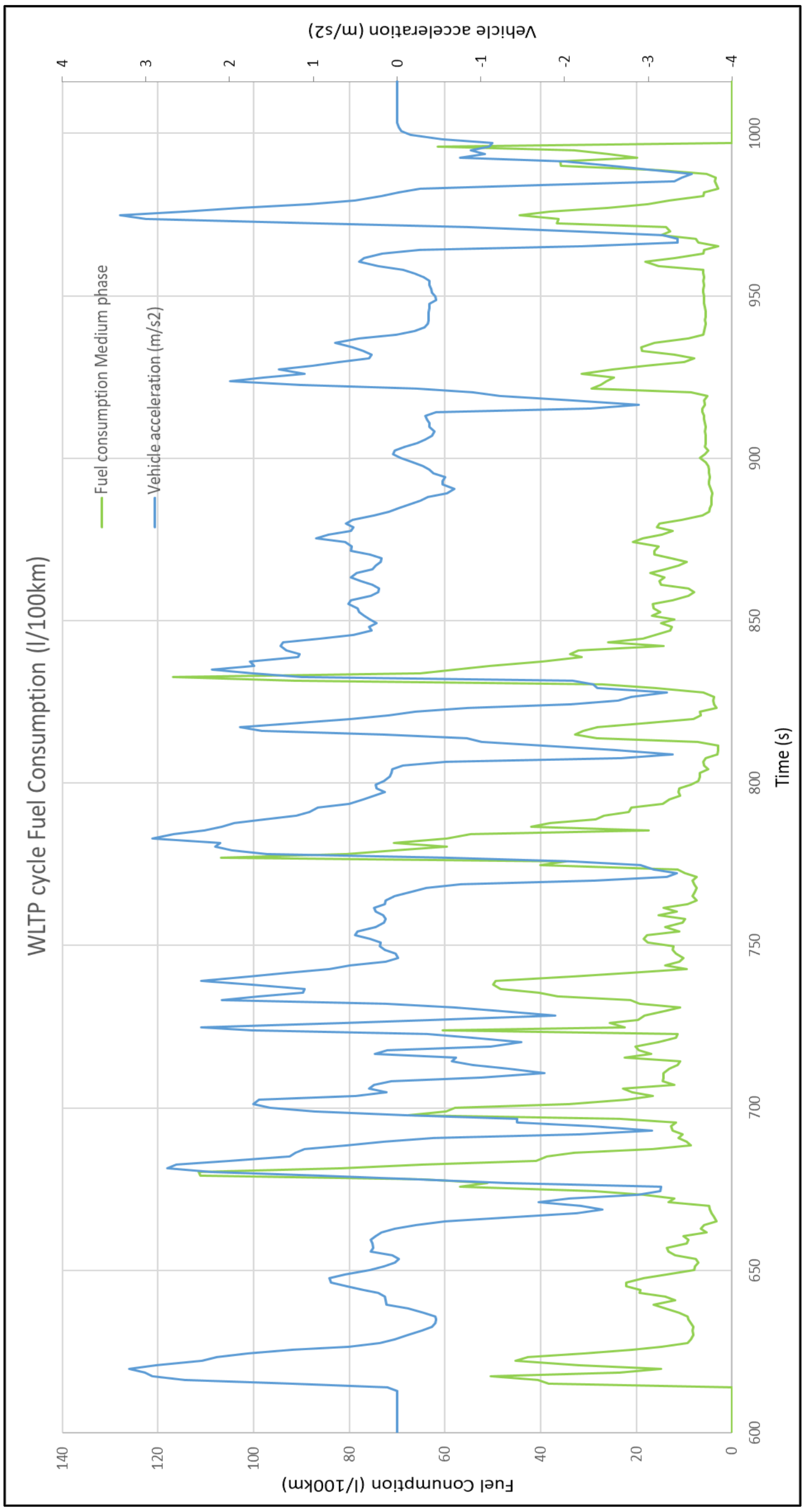


Fig. 53: Vehicle acceleration versus WLTP cycle fuel consumption.

The obtained fuel consumption over the WLTP cycle on a different phases according to values from dSpace HIL are shown in Table 7.

WLTP Cycle Phase	Fuel consumption (l/100km)
Low Phase	29.50
Medium Phase	19.04
High Phase	15.2
Extra High Phase	10.8
Overall Average	19.15

Table 7: Fuel consumption over the WLTP cycle different

8.2.2 Estimated CO_2 Emissions

The estimation of CO_2 is based on the fuel consumption complete combustion calculation assumption. And the results at different phases are as shown in table 8.

WLTP Cycle Phase	CO_2 emissions (g/km)
Low Phase	705.5
Medium Phase	455.5
High Phase	363.7
Extra High Phase	259.3
Overall Average	458.1

Table 8: Estimated CO_2 over the WLTP cycle different

As observed, the emissions also are completely dependent on the fuel consumption. Emissions are more in the phase where aggressive driving (more engine load phase) has occurred.

8.3 dSpace HIL Fault Code Diagnostics

As explained in the literature review chapters, a vehicle stores the trouble code in its memory when it detects a component or system that’s not operating within acceptable limits. The code will helps to identify and fix the issue within the vehicle.

To perform the TestCase smoothly without check engine light on, some of these error codes are masked out or put to the default value by the Porsche Engineers as it does not affect their tasks. The masking of the error are done using INCA Software, these code validate the results which is observed in the study of prediction of fuel consumption. Few of the relevant error codes which are masked/default valued are discussed along with its description in table 9:

FAULT CODE NAME	OBD DTC CODE	DESCRIPTION
IVGdiCtlCvoErrMax_x	P02CD,..	All Cylinder Fuel Injector
IVGdiCtlCvoErrMinMax	P0149	Fuel Timing Error
HEGOS2B1ElecNpl	P0136	O2 Sensor Circuit Malfunction
GEVlvPhaSlowIntkB1 and B2	P000A and P000C	Camshaft Position Slow Response Bank 1 and Bank 2
GEVlvPhaPsOpenLoadOutlB2	P0023	Camshaft Position Actuator Circuit (Bank2)
OilPPlaus	P0521	Engine Oil Pressure (EOP) Sensor
DFRMmax and DFRM2max	P0171 and P0174	Fuel Trim, System too Lean Bank 1 and Bank 2
DFRMmin and DFRM2min	P0172 and P0175	Fuel Trim, System too Rich Bank 1 and 2
SCtPmp2DiagGrdKeyErr	P261B	Coolant Pump, Control Circuit Range/Performance
FanErrNEngLim	P10EF	Function limitation due to failure of cooling fan
Fan1_KLERespErr	U0632	Lost Communication With Fan 1 and 2
ExhTSnsr4SentData	U14A1	Exhaust temperature bank 2 implausible message
PSRPmax	P0236	Turbocharger/Supercharger Boost Sensor
PSRPmax	P0106	Manifold Absolute Pressure Sensor
DSKVRmax	P0191	Fuel Rail Pressure Sensor Circuit Range/Performance
TWCDPriCatB1 and TWCDPriCatB2	P0420 and P430	Catalyst System Efficiency Below Threshold Bank 1 and 2

Table 9: HIL Fault codes and description

Chapter 9 Conclusion

9.1 Summary

The theoretical part of this thesis provides in-depth understanding of various concepts from Communication between In-Vehicle Networks, Vehicle Diagnostics, and lastly Hardware-in-the-Loop (HIL). To obtain optimal results and successfully implement the practical part, it is imperative to acquire the theoretical knowledge of these concepts.
For example, the topic of Vehicle Diagnostics is presented in detail to understand the meaning of the error and fault codes occurring in the vehicle memory during simulation results.

As one of the main objective of this work involves Hardware-In-Loop and driving cycles, this work provides a very clear understanding of requirements and functionality of the Porsche Engineering HIL setup.

The implementation methods of WLTP cycle are described using manual and EXAM TestCase (automation method).

Main conclusion

1. The TestCase was generated in EXAM with implementation of WLTP cycle.
2. This work also defined the various calculations and factors which are needed in estimating the fuel consumption and CO_2 emissions.
3. It has been verified that driving behavior affects the consumption, as random starts and stops present massive fuel consumption whereas normal driving without aggressive acceleration–deceleration provides good fuel economy.
4. Validation studies were carried out on HIL, using data collection from the real vehicle.

Reasons for HIL results over or underestimate by validation:

1. Modelled values of Short term fuel trim – From the results, the overestimate of fuel consumption is observed in case of HIL as short term fuel trim is unable to run HIL engine into lean region (λ=2) during engine braking phase. This effect the functioning of the Lambda sensor which influence the consumption.
2. Actuation of throttle valve – With the results, in all phases the throttle is not performing according to the real vehicle. This can be due to difference in the driving behavior of the driver model (real and HIL) or can be due to the modelled values in HIL. In the acceleration phase the HIL actuation from 80 to 40% is occurring in 0.5 sec while in real vehicle it's in 2.5 sec. This actuation can affect the engine load/MAF which plays a very significant role in the fuel consumption.
3. Intake Manifold Temperature of HIL – The inability of HIL to maintain the constant intake temperature in intake manifold after the turbocharger can also prove the reason

of the underestimation. The HIL values are in the range of 50°C - 60°C, while in real vehicle it is around 30°C, this variation can affect the density of the air which will further will effect fuel consumption. This effect can also be due to the improper modeling of the intercooler or the ambient temperature in the HIL.

4. Modeling of the gear shifting – According to the results 0 -120 Km/h real vehicle is on 2nd gear while HIL driver model is on 3rd gear, proves that the modeling of the gear shifting can be a problem.

Reasons for Real results over or underestimate by validation:

1. Grade (slope) on road – Measured results are performed on the Stuttgart, Germany roads and they have a little bit of slope on the road, this can affect the engine load.
2. Effect of Volumetric Efficiency (VE) – This effect played a very important role in determining the consumption of the real vehicle data. Due to absence of the MAF sensor values VE values are estimated according to the HIL values at different phases.
3. Effect of EGR – As MAF values of real vehicle are calculated using the speed density method, which assumes ignoring EGR. As the dilution effect of due to EGR is very important for the petrol air/fuel mixture combustion, this also influence fuel consumption.
4. Effect of Coolant temperature: This effect is also related to the Exhaust gas recirculation (EGR) which in turn improve the fuel consumption. But it observed that the dSpace HIL is not varying the temperature values, the HIL engine coolant temperature is set on default at 88°C while the real vehicle is running between 94°C - 97°C. This can also be due to the improper modeling of the coolant pump and cooling fan.

9.2 Contribution of thesis

- A test case in EXAM was generated that use WLTP cycle to validate HIL system focusing on fuel consumption.
- This TestCase can potentially help the future engineers working on this system to test the vehicle using WLTP cycle and advance their work more timely and reliably.
- Moreover, in future, using this study, calculations and development for estimation of fuel consumption, the emissions model for the HIL can be generated which can further be validated using real values from exhaust analyzers fitted on the vehicle.
- Also, WLTP cycle real results can be used to validate the HIL system in future.
- Using the fault code diagnostic analysis of HIL, in future it can be resolved and the performance of the Hardware-in-loop can be improved.

References

[1] Will Electric Cars Transform the U.S. Car Market? by Lee, H. et. Al, Harvard Kennedy School, Belfer Center for Science and International Affairs, July 2011.

[2] Enhancing Automotive Embedded Systems with FPGAs, Shreejith Shanker.

[3] In-Vehicle Networking, B.K.Ramesh, K. Srirama Murthy, Dearborn Electronics.

[4] ISO 11898-1-Road vehicles – Controller Area Network (CAN) – Part 1: Data link layer and physical signaling, 2003.

[5] ISO 15765-2 - Road vehicles – Diagnostics on CAN – Part 2: Network layer services.

[6] Wikipedia, CAN bus — Wikipedia, the free encyclopedia.

[7] ISO 17458-2 Road vehicles — FlexRay communications system — Part 2: Data link layer specification, 2013.

[8] ISO 10681-2 Road vehicles — Communication on FlexRay — Part 2: Communication layer services, 2010.

[9] ISO 17987-3 Road vehicles — Local Interconnect Network (LIN), 2016

[10] Introduction to the Local Interconnect Network (LIN) Bus, National Instruments, 2019

[11] Ethernet as Future Automotive Communication Backbone by Rolf Ernst, Philip Axer, Daniel Thele, Technische Universitat Braunschweig, Germany

[12] The Multi-Domain Frame Packing Problem in CAN-FD by Prachi Joshi, June 2017

[13] Automotive Ethernet by Vector Informatik GmbH, 2020.

[14] In-Vehicle Execution Environment for Diagnostic Scripts on Heavy Commercial Vehicles by TOBIAS GUSTAFSSON, KTH Sweden, 2016

[15] Wikipedia, On-board diagnostics — Wikipedia, the free encyclopedia [accessed 14-March-2020].

[16] EUR-Lex - 31998L0069 - EN. European Parliament, 1998.

[17] SAE J1962: Diagnostics Connector Equivalent to ISO/DIS 15031

[18] SAE J1979 / ISO 15031-5 Diagnostic Test Modes [accessed 15-March-2020]

[19] Wikipedia, OBD-II_PIDs— Wikipedia, the free encyclopedia [accessed 15-March-2020]

[20] ISO 14229-1: Road vehicles — Unified diagnostic services (UDS)

[21] UDS Message Structure, https://embedclogic.com/uds-protocol/uds-frame-type/

[22] INCA Software products, https://www.etas.com/en/products/inca_software_products-details.php

[23] Computerized engine controls 11th edition book, by Steve V. Hatch

[24] What is a VW EPC warning light, https://volkswagen-polo-highline.blogspot.com/

[25] Hardware-in-the-Loop: The Technology for Testing Electronic Controls in Vehicle Engineering, Dr. Peter Waeltermann dSPACE Inc. 6th Paderborn Workshop on Designing Mechatronic Systems, 2009

[26] Hardware-in-the-Loop Testing in the Context of ISO 26262, Andreas Himmler, Klaus Lamberg and Michael Beine, dSPACE GmbH

[27] DiagRA® D: The diagnostics tool for developers: Version 7.41.40.42644 by RA Consulting GmbH, 2020

[28] EXAM Modeler training hardcopy by Micronova Software und Systeme

[29] Development of a World-wide Worldwide harmonized Light duty driving Test Cycle (WLTC) technical report by Monica Tutuianu et al, United Nations Economic Commission for Europe, 2013

[30] Essentials of Vehicle Dynamics by P.Pauwelussen, 2015

[31] Engine Breathing - Steady Speed Volumetric Efficiency and Its Validity Under Transient Engine Operation by L. A. Smith et al, SAE International in United States, 1999

[32] Spalovací motory Komplexní přehled problematiky pro všechny typy technických automobilních škol by Jan Hromádko, 2011.

[33] Theory of Internal combustion engine lecture presentations by doc. Ing. Oldřich Vítek, CVUT, 2019.

[34] Internal Combustion Engine Fundamental by John B Heywood, McGraw Hill.

[35] Premixed Combustion in Spark Ignition Engines and the Influence of Operating Variables by Fabrizio Bonatesta, 2012.

[36] Estimation of engine intake air mass flow by Michal Vojtíšek, Czech Technical University, Prague, 2014

[37] Exhaust Gas Recirculation (EGR) complete guide – introduction by x-engineer.org

[38] Exhaust gas recirculation boosted direct injection gasoline engines by A. Cairns et al, MAHLE Powertrain Ltd, UK, 2009.

[39] Engine Coolant Temperature (ECT) Sensor – Function – Failure And Testing by Danny Bender, https://dannysengineportal.com/

[40] Linking Received Packet to the Transmitter Through Physical-Fingerprinting of Controller Area Network by Omid Avatefipour, et al, University of Michigan, 2018.

[41] Next Generation Car Network- FlexRay by Fujitsu Microelectronics (Shanghai) Co., Ltd, 2006

Appendix

CONTROLLER AREA NETWORK (CAN)

FIELD NAME	LENGTH (BITS)	PURPOSE
Start of frame	1	Denotes the start of frame transmission
Identifier (green)	11	A (unique) identifier which also represents the message priority
Remote Transmission Request (RTR) (blue)	1	Must be dominant (0) for data frames and recessive (1) for remote request frames
Identifier Extension Bit (IDE)	1	Must be dominant (0) for base frame format with 11- bit identifiers
Reserved bit (r0)	1	Reserved bit. Must be dominant (0) but accepted as either dominant or recessive.
Data length code (DLC) (yellow)	4	Number of bytes of data (0–8 bytes)
Data Field (red)	0-64 (or 0-8)	Data to be transmitted (length in bytes dictated by DLC field)
CRC	15	Cyclic redundancy check
CRC delimiter	1	Must be recessive (1)
ACK slot	1	Transmitter sends recessive (1) and any receiver can assert a dominant (0)
ACK delimiter	1	Must be recessive (1)
End of Frame (EOF)	7	Bit indicates the end of the messages and disables the bitstuffing.

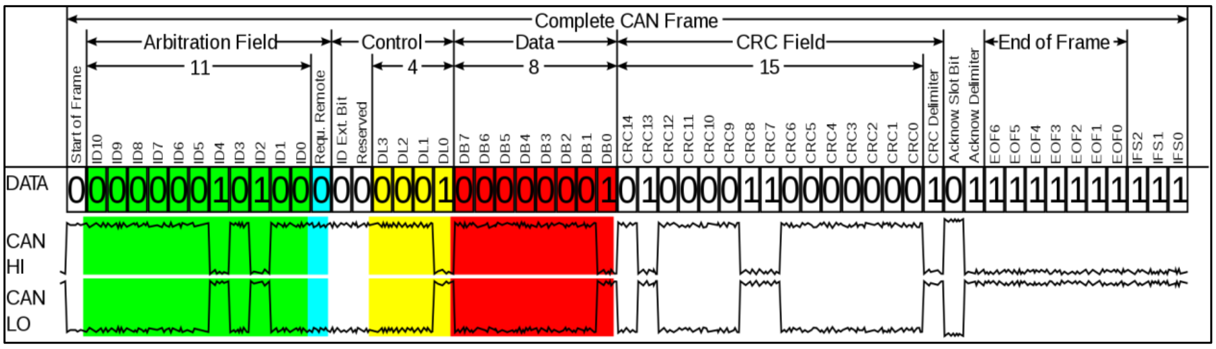


Fig. 54: CAN-Frame in base format (11

The ISO-11898:2003 standard was originally created for messages with an 11-bit identifier (ID). The standard was later amended with the extended 29-bit identifier. The 11-bit IDs provide 2048 valid different messages identifiers, where the 29-bit IDs provide room for 537 million possible message identifiers.

CAN Bus Signal

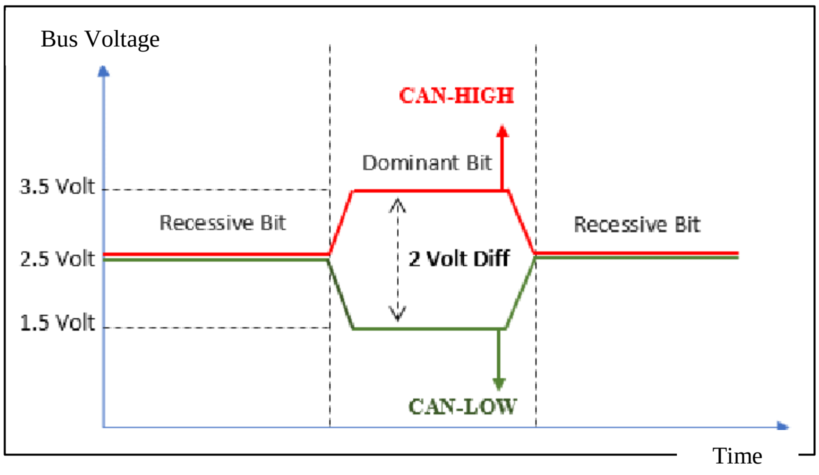


Fig. 55: CAN bus levels [40]

On the physical level, a CAN bus exists out of a twisted pair where in rest 2,5V is applied on in case for high-speed variant. When a node wants to send a recessive bit (associated with a logic 1), the voltage of 2,5V is not changed. When a dominant bit has to be sent (associated with a logic 0), one wire of the twisted pair is pulled to 1.5V (the CAN Low wire) and the other one is lifted to 3.5V (the CAN High wire) in case of high speed variant. Both wires are closed with a 120Ω resistor to suppress reflections on the bus (Figure 55). The nodes have to ensure their own synchronization to the messages on the bus. As CAN is event-triggered communication and not time-triggered therefore, no clock will be transmitted with the CAN messages.

FLEXRAY

Multi-drop Bus

The Multi-drop Bus topology is the same as in CAN and LIN systems, and it uses a single bus to connect multiple processors together. This system is helpful in implementing with other systems due to the similar layout of CAN and LIN networks.

Star Network

The Star Network topology consists of multiple ECUs connected to a central active node. This layout is useful in that if one processor is cut or disconnected, the other processors continue functioning.

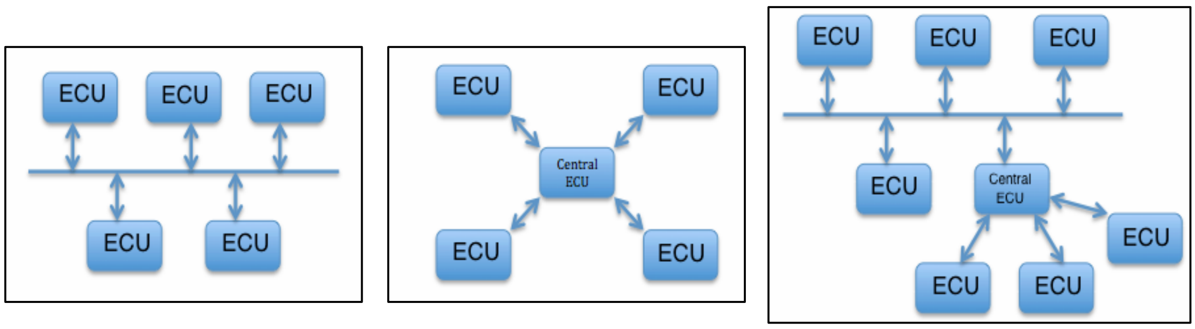


Fig. 56: FlexRay Hybrid

Hybrid Network

The Hybrid Network topology is a mix between Multi-bus and Star Networks. This layout combines the reliability and cost efficient advantages of the other two layouts, and it will most likely be the future of FlexRay technology as shown in the Figure 56 above.

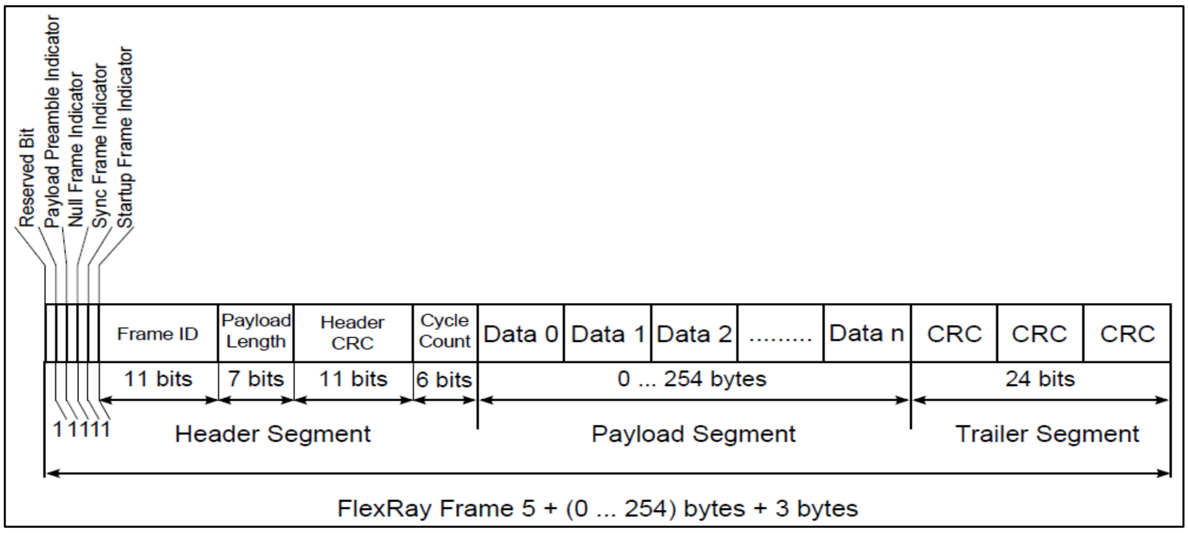


Fig. 57: FlexRay Frame Format

FIELD NAME	LENGTH (BITS)	PURPOSE
Reserved bit	1	This bit is not currently used by the protocol and has been reserved for further use.
The payload preamble indicator	1	This bit specifies the existence of the vector information in the payload segment of the frame. In the static frame it indicates a Network Management Vector and in a dynamic frame it indicates message ID.
Null frame indicator	1	This bit designates whether or not the frame is a null frame i.e. a frame that contains no usable data in the payload segment of the frame.
Sync frame indicator	1	This bit designates whether or not the frame is a sync frame i.e. a frame that is utilized for system wide synchronization of communication.
Startup frame indicator	1	Shows whether or not the node sending frame is the start-up node.
Frame ID	11	The frame ID defines the slot in which the frame should transmitted and is used for prioritizing event-triggered frame. Within the same time slot the node can transmit frames with different IDs. Valid frame IDs range from 1 to 2047.
Payload length	7	This stated the data length of the payload segment.
Header CRC	11	This is the CRC calculation value of sync frame indicator, startup frame indicator, frame ID and payload length which is calculated by the host.
The cycle count	6	This indicates the value of the cycle counter on the transmitting node at the time of the frame transmission.

FlexRay Signal

At the physical layer, FlexRay communicates using the differential signals BP and BM, corresponding to the voltages uBP and uBM (Figure 58).
The differential voltage between the signals (V_{diff}) is due to represent four different sates which can occur on the bus: Idle_LP: low-power state, Idle: no-communication state, Data_1: logical HIGH and Data_0: logical LOW

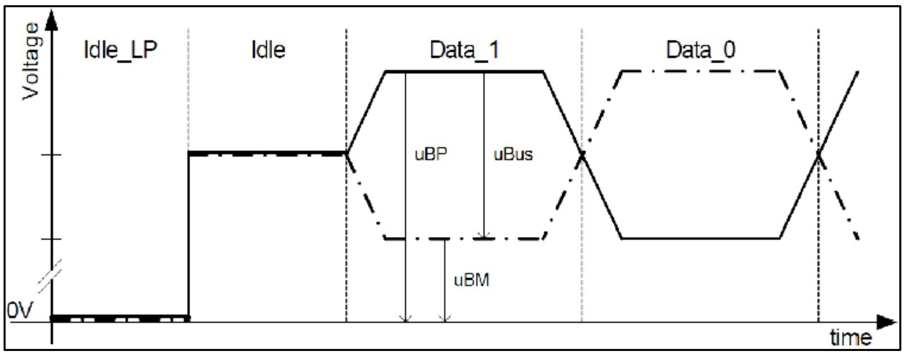


Fig. 58: FlexRay Signal [41]

LOCAL INTERCONNECT NETWORK (LIN)

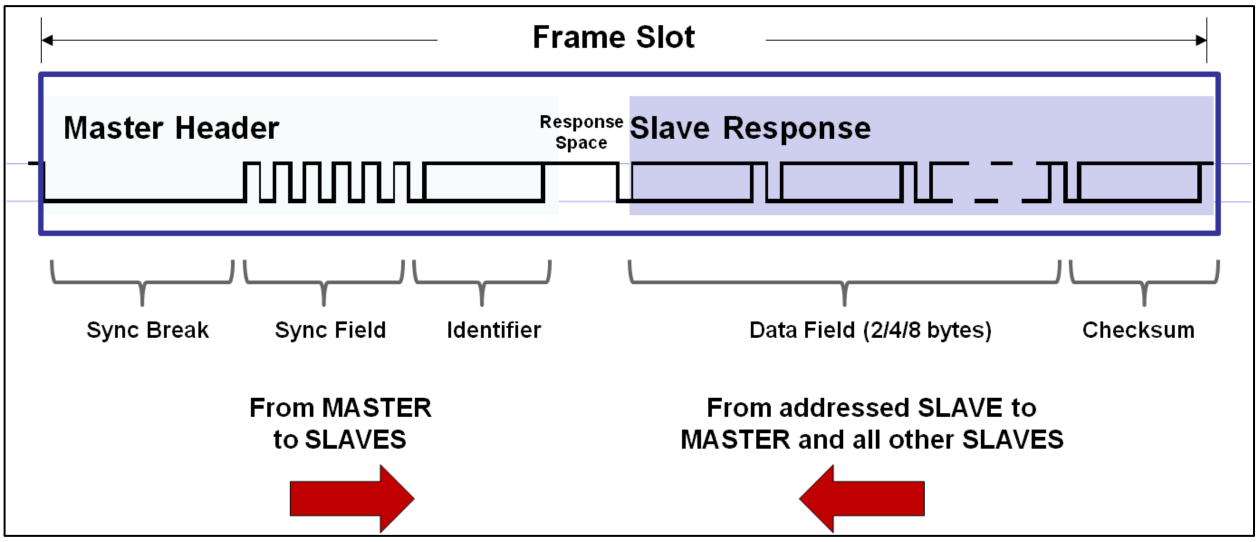


Fig. 59: LIN Frame Format

Master Header

Sync Break: Every LIN frame begins with the break, which comprises 13 dominant bits (nominal) followed by a break delimiter of one bit (nominal) recessive. This works as a start-of-frame notice to all nodes on the bus.

Sync Field: The master task in the header transmit the sync field as a second field. Sync is well-defined as the character x55. The sync field permits slave devices that perform automatic baud rate detection to measure the period of the baud rate and adjust their internal baud rates to synchronize with the bus.

Identifier: This field is the final field transmitted by the master task in the header. It provides identification for each message on the network and ultimately decides which nodes in the network receive or respond to each transmission. All slave tasks repeatedly listen for ID fields, verify their parities and determine if they are publishers or subscribers for this specific identifier. The LIN bus consist of a total of 64 IDs. IDs 0 to 59 are intended for signal-carrying (data) frames, 60 and 61 are used to transfer diagnostic data, 62 is kept for user-defined extensions and 63 is reserved for future protocol developments. The Identifier is communicated over the bus as one protected ID byte, with the lower six bits consisting of the raw ID and the upper two bits comprising the parity.

<u>Slave Response</u>

Data Bytes: These bytes field are transmitted by the slave task in the response. This field comprises from one to eight bytes of payload data bytes.

Checksum: This field is transmitted by the slave task in the response. The LIN bus states the use of one of two checksum algorithms to calculate the value in the 8 bit checksum field. Classic checksum is computed by adding the data bytes alone and enhanced checksum is computed by adding the data bytes and the protected ID.

<u>LIN Signal</u>

It was the goal of the LIN design to accomplish a simplistic wiring topology. The simple single-wire bus connects to each node in the collection and switches from ground to battery-level voltage as shown: Signal Levels (Figure 60)

- Dominant - Bus LOW - Logic 0
- Recessive - Bus HIGH - Logic 1

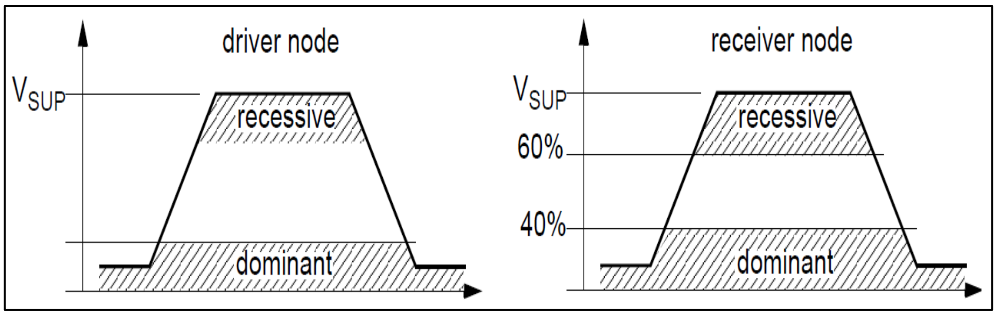


Fig. 60: LIN Signal