POLITECNICO DI TORINO

Master Degree in Computer Engineering
Embedded Systems

Master Degree Thesis

Test of Automotive Boards

Study and implementation of testing algorithms for ECUs using AUTOSAR standard

Advisor
Matteo Sonza Reorda

Co-Advisor:
Massimo Violante

Candidate
Andrea Santeramo

Internship Tutor - Magneti Marelli
Testing & Tools Engineering
Paolo Caruzzo

Academic Year 2018 – 2019

. . . .
Andrea Santeramo

Turin, September 2019

Summary

The ever increasing Electric/Electronic (E/E) complexity of automotive systems was a major motivation which led to the development of the AUTomotive Open System ARchitecture (AUTOSAR) standard.

A modern car may have more than hundred Electronic Control Units (ECUs) and each of them contains thousands of functions which must often be rewritten from scratch when hardware is changed. It was urgent need for automotive giants to cooperate together in order to make application Software (SW) independent from Hardware (HW). In order to accomplish this, basic functions are implemented in AUTOSAR as an industry wide standard core solution and to make software modular and reusable.

The subject of this thesis work is the study of an engine control ECU. All the advantages and disadvantages that led to the migration to the AUTOSAR standard were taken into consideration. The implementation strategies and the division of the basic components into the new layered structure that this standard provides have been studied.

Particular attention has been devoted to the *testing* aspect, which is the specific of this thesis.

The test strategies in the production flow impacting on the hardware fault coverage are described, highlighting for each technique the general theoretical concepts in form of equipment involved, potentialities, limits and possible future improvements.

Acknowledgements

Special thanks to my parents, for letting me get here . . .
to my friends, for supporting me along the way . . .
and . . . least but not last
thanks to my lovely girl ♡ for having been beside me during this winding path.

“Stay hungry,
stay foolish”
Steve Jobs

Contents

Glossary

AC Alternating Current. 18–21

ADAS Advanced Driver Assistance Systems. 24, 25

ADC Analog Digital Converter. 34

AOI Automated Optical Inspection. 4, 55, 56

API Application Programming Interface. 33, 35, 36

ASIL Automotive Safety Integrity Level. 4, 21, 54

ATE Automatic Test Equipment. 15, 57–61

AUTOSAR AUTomotive Open System ARchitecture. iii, 3, 4, 29–33, 35–37, 42, 43, 70–73, 77, 78

BCA Basic Control Algorithms. 41

BEV Battery Electric Vehicle. 3, 12, 17, 18, 50, 51

BMS Battery Management System. 3, 20, 21, 27

BSW Basic Software. 32, 33, 36, 37, 43, 72

CAN Controller Area Network. 5, 24, 34, 43–45, 67, 68, 70, 78

CSMA/CR Carrier Sense Multiple Access/Collision Resolution. 24

DC Direct Current. 18–21

DCM Diagnostics Communication Manager. 4, 36, 37

DD Device Driver. 67–69, 73, 78

DDI Device Driver Interface. 41

DEM Diagnostics Event Manager. 37

DID Data Identifiers. 36

DIO Digital Input/Output. 34

DOD Depth of Discharge. 20

DSP Diagnostic Service Processing. 36

DTC Diagnostic Trouble Code. 35, 37

DUT Device Under Test. 4, 15, 55–61

E/E Electric/Electronic. iii, 31, 54

ECU Electronic Control Unit. iii, 3–5, 11–13, 23–26, 31–37, 39–44, 51–53, 59, 60, 65–68, 70–73, 77, 78

EDD ECU Device Driver. 40

EEPROM Electrically Erasable Programmable Read-Only Memory. 14, 35

EMC Electromagnetic Compatibility. 25

EMF Electromotive Force. 25, 53

EMI Electromagnetic Interference. 51

EMS Engine Management System. 11, 12

EOL End of Line. 3, 4, 14, 58–61, 77

EPROM Erasable Programmable Read-Only Memory. 41

EV Electric Vehicle. 19–21

FIM Function Inhibition Module. 37

FLS Flash. 34

FW Firmware. 11, 13–15, 60, 65–69, 77

GDI Gasoline Direct Injection. 12

GPT General Purpose Timer. 33

HEV Hybrid Electric Vehicle. 3, 12, 17, 19–21, 50, 51

HIL Hardware in the Loop. 4, 26, 51–53

HV High Voltage. 17–20

HW Hardware. iii, 13–15, 21, 31, 49, 54, 58, 60, 65–67, 69, 72, 73, 77, 78

ICE Internal Combustion Engine. 3, 17–19

ICT In-Circuit Test. 4, 14, 56–58, 61

ICU Input Capture Unit. 34

IGBT Insulated Gate Bipolar Transistor. 19, 20

ISO International Organization for Standardization. 21, 36, 43, 68

KERS Kinetic Energy Recovery System. 8

LIN Local Interconnection Network. 24, 34

LV Low Voltage. 17, 19

Magneti Marelli Magneti Marelli S.p.A.. 3, 5, 7–9, 11–13, 23, 39, 40, 42, 65, 67, 70, 73

MBT Model Based Testing. 4, 49

MCAL Microcontroller Abstraction Layer. 32, 33, 71–73

MCU Microcontroller Unit. 33, 34, 50, 51, 54, 59, 61, 66, 72

MOSFET Metal-Oxide Semiconductor Field-Effect Transistor. 20

MOST Media Oriented Systems Transport. 24

NM Network Management. 42

OBD On-Board Diagnostics. 4, 36

OBT On Board Test. 59, 60

OEM Original Equipment Manufacturer. 31, 42, 51

OS Operating System. 35, 66

OSEK Open Systems and their Interfaces for the Electronics in Motor Vehicles. 42

PCB Printed Circuit Board. 56, 57

PCBA Printed Circuit Board Assembly. 55, 56

PFI Port Fuel Injection. 12

PIL Processor in the Loop. 4, 50, 51

PRI Power Run-In. 14

PWM Pulse Width Modulation. 5, 33, 34, 52, 67, 69

PWT Powertrain. 9, 12, 13, 15, 18, 40, 42

R&D Research and Development. 8

RTE Runtime Environment. 32, 35, 43, 72

SCM Software Configuration Management. 65

SDC Standard Digital Core. 40

SID Service Identifiers. 36

SIL Software in the Loop. 4, 50

SOC State of Charge. 20, 21

SOH State of Health. 20, 21

SPI Serial Peripheral Interface. 33, 34, 41

SUT System Under Test. 49

SW Software. iii, 3–5, 14, 21, 31, 39, 49, 65, 67, 70, 71, 77

SWC Software Component. 30, 31, 35, 43, 49, 71–73

TDMA Time Division Multiple Access. 24

TPGM Test Program. 14

TTE Testing & Tools Engineering. 3, 12, 13, 15, 65, 66

UDS Unified Diagnostic Services. 4, 5, 35, 36, 43, 44, 67–70, 78

VFB Virtual Functional Bus. 30, 31

WDG Watchdog. 34, 35

Part I

INTRODUCTION

Chapter 1

Introduction

This thesis work analyzes the testing techniques for automotive ECUs in the End of Line (EOL) phase. The SW architecture used today and the future developments based on the AUTOSAR standard are therefore analyzed by studying their features, advantages and disadvantages. ECU testing methods with different SWs are then compared, by analyzing their characteristics, complexity and potential. This work ends with the analysis of an ECU based on AUTOSAR standard, and the study and implementation of test modules.

Before talking about ECU, the company that hosted and supported me during this thesis work deserves particular regards. This study started by going into detail on the powertrain sector, and on the Battery Electric Vehicles (BEVs) in which the amount of electrical and ECU signals inside the vehicle is greatly increased with respect to a traditional vehicle with an Internal Combustion Engine (ICE) powertrain.

The chapter 2 introduces the Magneti Marelli S.p.A. (Magneti Marelli) history, starting from the origins (section 2.1.1) up to nowadays (section 2.1.2), including an overview of the new company Marelli (section 2.1.3). The main company business lines are also analyzed in this part (section 2.2).

The chapter 3 describes what the powertrain is, and it talks about the Magneti Marelli Powertrain business line products (section 3.1). It also shows the composition of the Testing & Tools Engineering (TTE) team that is in charge of testing powertrain products. In this context, the interactions of the test engineer figure with respect to the other teams in the company hierarchy is very important (section 3.2).

The chapter 4 focuses on BEVs and Hybrid Electric Vehicles (HEVs) whose importance is growing exponentially today. The new concept of powertrain is then analyzed since here the fuel is replaced by a battery pack (section 4.1). In the following section it is explained more in detail the Battery Management System (BMS) (section 4.4) and the complexity of the electronics used to control the charge/discharge of the battery pack.

The chapter 5 explains in detail what an ECU is, what are the main functions of an ECU, and why it is so important to make sure that the device will not fail during its work. Errors in ECUs are not allowed, and they should be avoided by testing the whole device before sending it to the customer. The main methods for ECU wire communication are also described (section 5.1) by underlining the main advantages and disadvantages. Then, knowing what an ECU does and what functions it must perform, it is explained what has to be tested (section 5.2). In particular, the functions and the signals that have to be tested are analyzed in order to be sure that the device works as intended. Testing involves the stimulation of specific hardware pins in order to verify their correct behavior. Often it becomes necessary to recreate the working conditions of the ECU to be able to test its functioning.

The chapter 6 introduces the AUTOSAR standard starting from its history (section 6.1) in order to better understand the historical reasons that carried some big automotive giants to join together in such consortium. Then, the main generalities of the standard are described in detail (section 6.2), up to the detailed description of the layered architecture that provides the standard, highlighting the key points (section 6.3). A focus is done on the vehicle diagnostic features that the AUTOSAR standard offers, underlying the main characteristics and functions of the Diagnostics Communication Manager (DCM). This module can read/write ECU errors, and it also supports the diagnostic protocols Unified Diagnostic Services (UDS) (ISO 14229) and On-Board Diagnostics (OBD) II (section 6.4).

The chapter 7 describes the architecture of an ECU in the host company portfolio. Different SW architectures are presented: the oldest one (section 7.1), the currently used one (section 7.2), and the new AUTOSAR-based architecture for which a complex job of searching for analogies and change of implementation strategies was necessary in order to fully exploit the potential that AUTOSAR offers (section 7.3). Testing techniques are analyzed (section 7.3.1). The UDS protocol is then described (section 7.4), since it is the most used protocol for diagnostic purpose, and it is used for testing purpose too by the company through a specific module for ECU communication (section 7.5).

The chapter 8 describes the development of test automation for the automotive system model (Model Based Testing). The Device Under Test (DUT) can be tested in different scenarios, and models can be used to represent the desired behavior of the DUT. There is a focus on the Software in the Loop (SIL) (section 8.2), Processor in the Loop (PIL) (section 8.3), Hardware in the Loop (HIL) (section 8.4) testing techniques. The following chapter explains the safety standard as proof that substantial safety measures have been taken for a certain product (section 8.5). There is also a standard ISO 26262 (section 8.5.1) that ensure functional safety to "absence of unreasonable risk due to hazards caused by malfunctioning behavior", and the different safety levels Automotive Safety Integrity Level (ASIL) are explained in order to distinguish different safety protection levels among.

The chapter 9 describes the steps through which the bare board is transformed into a final product passing through Front-End and Back-End phases. Some potential defects can be introduced by the product due to imperfections of the production process. The responsibility of a test line is to achieve the highest capability in defects-detection, also called Hardware Fault Coverage. This chapter focuses on the test strategies impacting on the final coverage: Automated Optical Inspection (AOI) (section 9.1), In-Circuit Test (ICT) (section 9.2) and EOL (section 9.3).

The chapter 10 summarizes the work done in the company starting from the description of the programs and tools used (section 10.1), then showing some examples of function calls to test hardware pins. The test code writing phase is obviously followed by a compilation and subsequent loading on the target in such a way as to verify the behavior of the device (section 10.2). In particular, the proprietary Magneti Marelli SW architecture (section 10.3) is analyzed with regard to the ignition of a Pulse Width Modulation (PWM) output pin (section 10.3.1). This operation is performed again by using the Controller Area Network (CAN) protocol, and a UDS message as means to communicate the ignition command of the same PWM output pin to the ECU (section 10.4).

The chapter 11 collects personal reflections and comments on the work done. The last chapter emphasizes the motivations and the reasons that led the company to make the choice to write test programs for the ECUs. The goals set and the results obtained are compared. Possible further research developments are then analyzed.

Chapter 2

Magneti Marelli

2.1 History

This is a brief introduction to fully understand the Magneti Marelli position in the world. Starting from what it was, up to what this company is today. What are the main fields of application, and in which contexts is the figure of a *test engineer* in a company like this.

2.1.1 The past

In 1891, Ercole Marelli founded the company bearing his name, specialized in the production of electrical devices and engines. After about 25 years, Fiat and the "Società anonima Ercole Marelli", in order to satisfy the growing mobility sector demand, established a partnership to start a mass production of magnets for internal combustion engines. Magneti Marelli was founded on 8th October 1919 in equal parts by Fiat and "Società anonima Ercole Marelli". Their activities, from 1920, became a huge variety so that a in-house personnel training was launched to prepare employees on the study and design of new products related to automobiles, motorbikes, industrial motors, aviation and racing engines. An important contribute of Magneti Marelli was related to the production of professional radio equipment for land, aeronautical and naval communications, in 1930. Five years later, a partnership with Bosch was established for the marketing of electrical equipment for automobiles and motorbikes. Thanks to the acquired popularity, Magneti Marelli was chosen to test, design and build the first television broadcasts for RAI, from the camera to the TV, in 1939. In 1940 Magneti Marelli became also a supplier to the Italian Army, Navy and Air Force, for which it produced electrical equipment and mobile reception/transmission systems. An important breakthrough was in the 1960s, when the Magneti Marelli activities gradually started to focus on the automotive industry. In fact, in 1963, the entire telecommunication sector was sold in order to acquire a company specialized in the manufacture of ignition spark-plugs but also to inaugurate the first automotive production facility in Brazil, in 1978 (one of the main hubs of the current company's activities). The study and production of electronic control devices for ignition and supply systems was definitively set up after the foundation of Marelli Autronica (joint venture between Magneti

Marelli, Fiat and Weber). In 1980s, the entire sector of automotive batteries and accumulators was completely controlled by Magneti Marelli which, thanks to the acquisition of some other companies, became a leader in the field of lighting, in-vehicle electronics and fuel control/supply systems. The process of focusing only on the automotive field fully completed definitively in the 1990s, when strategies were adopted to strengthen the automotive core business and expanded it on the international markets. In the 1996, in fact, Magneti Marelli started its operation in China with the Guangzhou plant and some additional production facilities in Shanghai [1].

2.1.2 Nowadays

With the start of the new Millennium, the company expanded its vocation of major components manufacturer being able to design and produce complete automotive systems for leading car-makers. In 1998, it started its activities in the area of satellite navigation. Between 2000 and 2001, Fiat decided to sell some of their business branches such as electronic systems and air-conditioning. The same year, Magneti Marelli took full control of Automotive Lighting previously a joint venture with Bosch. An important contribute to the environmental sustainability, safety and "intelligent" automobiles was given by Magneti Marelli in 2003, when it launched, in Brazil, a combustion engines able to run both on ethanol, gasoline or any blend of the two. In 2009, after 90 years from its foundation, Magneti Marelli, helped by Fiat and Confindustria, extended its perimeter in the whole India and Shanghai. In the same years, it also developed the Kinetic Energy Recovery System (KERS), a system used in Formula 1 for the energy recovery under breaking. Nowadays, Magneti Marelli is extended in 19 countries and counts 43,000 employees, 86 production centers and 14 Research and Development (R&D) centers [1].

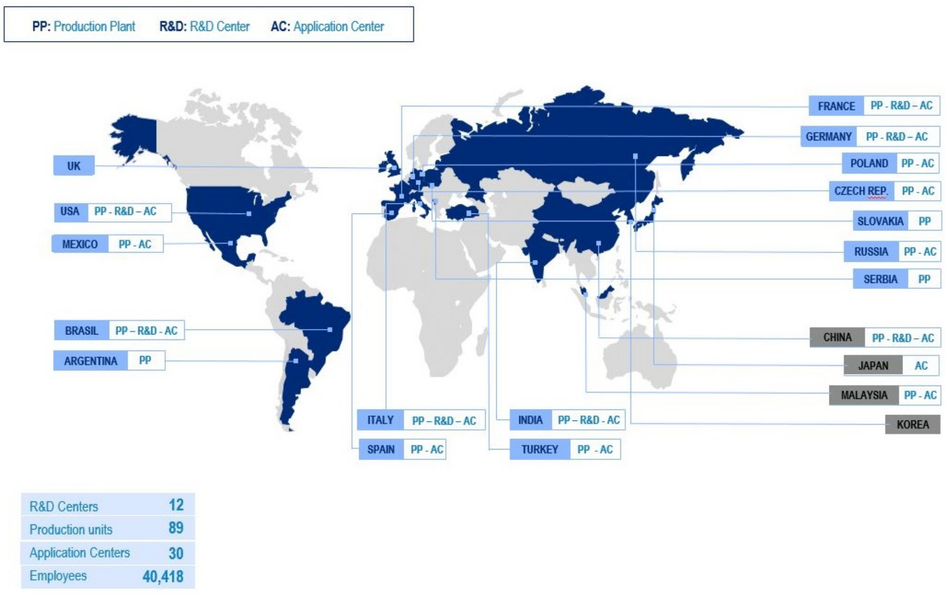


Figure 2.1. Magneti Marelli Worldwide presence

2.1.3 Marelli

Calsonic Kansei and Magneti Marelli recent joined together under a new global brand: **"MARELLI"** as part of the strategy of the new joint company to compete on a global scale. MARELLI combines two successful global automotive suppliers, one Italian, the other Japanese, both internationally recognized for innovation and manufacturing excellence.
MARELLI will be able to count on around 170 plants and research and development centers, in Europe, Japan, America and Asia-Pacific, and will have operational offices in Saitama (Japan) and Corbetta (Italy) [2].

2.2 Business Line

Magneti Marelli develop intelligent systems and solutions that contribute to the advancement of mobility, according to environmental sustainability, safety and quality of life on-board the vehicles.
The company is divided into these main business lines:

- Automotive Lighting (front and rear lighting systems)
- *Powertrain (PWT)* (engine control systems for gasoline, diesel and multi-fuel engines; Automated Manual Transmission "Freechoice" gearboxes)
- Electronic Systems (instrument clusters; infotainment & telematics, lighting & body electronics)
- Suspension Systems and Shock Absorbers (suspension systems, shock absorbers. dynamic systems)
- Exhaust Systems (exhaust systems, catalytic converters and silencing systems)
- Plastic Components and Modules
- After Market Parts and Services (Spare Parts for the Independent Aftermarket)
- Motorsport (electronic and electro-mechanical systems specifically for championships at the cutting edge of technology, in F1, MotoGP, SBK, WRC) [3].

A particular attention is related to the Powertrain (PWT) business line, in which the thesis was carried out. The next chapter introduces the PWT business line and describes all the aspects concerning the team in charge of testing PWT products.

Chapter 3

Powertrain

In a motor vehicle the powertrain comprises the main components that generate power and deliver it to the road surface, water, or air. This includes the engine, transmission, drive shafts, differentials, and the final drive (drive wheels, continuous track as in military tanks or caterpillar tractors, propeller, etc.). More recently in hybrid powertrains the battery, the electric motor and the control algorithm are also seen as elements of the powertrain.
In a wider sense, the powertrain includes all the components used to transform stored (chemical, solar, nuclear, kinetic, potential, etc.) energy into kinetic energy for propulsion purposes. This includes the utilization of multiple power sources and non–wheel-based vehicles.
The most recent developments in powertrain are driven by its electrification in multiple components. Electrical energy needs to be provided, and usually this leads to larger batteries. Electrical engines can be found as isolated components or as parts of other elements. In hybrid powertrains the torque generated by the combustion engine and the electric motor have to be brought together and distributed to the wheels. The control of this process can be quite involved but the rewards are greatly improved acceleration and much lower emissions [4].

3.1 Magneti Marelli Powertrain Products

The Magneti Marelli Powertrain is the business line dedicated to production of engines and transmissions for cars, motorbikes and light vehicles.

The figure 3.1 shows an Engine Management System (EMS). From left to right, the ECU, which runs a Firmware (FW), detects the acceleration driver intent. According to this, it actuates the combustion engine which, in turn, converts combustion to powertrain for the vehicle. It is important to notice that there are many sensors that help the ECU in this process, precisely to understand what is happening in the combustion engine during actuation. This process works thanks to the cooperation of several components.

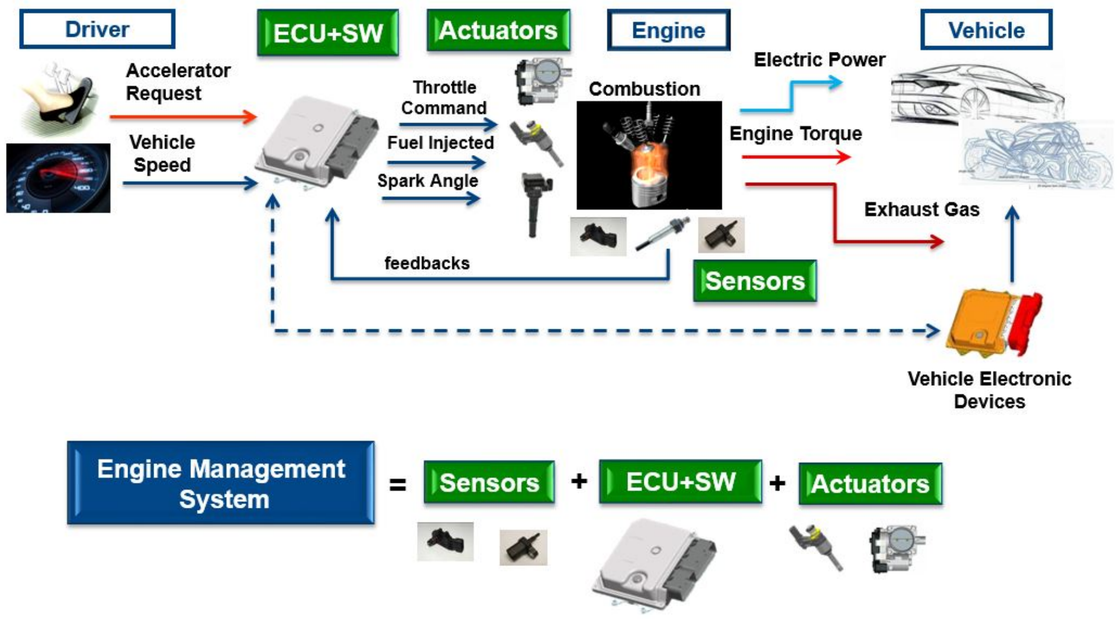


Figure 3.1. Engine Management System

In particular, Magneti Marelli-PWT produces the parts of the mechanism which composes the EMS. This production refers to the market of:

- Gasoline Systems - Gasoline Direct Injection (GDI)
- Gasoline Systems - Port Fuel Injection (PFI)
- HEV
- BEV
- Transmission
- Motorbikes
- Multi-fuel Systems.

Battery Electric Vehicles will be analyzed more carefully later because they are the subject of this thesis. In particular, the ECU used for my tests is for an electric vehicle [5].

3.2 Testing & Tool Engineering Department

This thesis was carried out within the TTE department. In particular, I worked closely with the team in charge of testing the functionalities of electronic parts of ECUs.
As figure 3.2 shows, the production of Electronic Components within the PWT business line involves different teams:

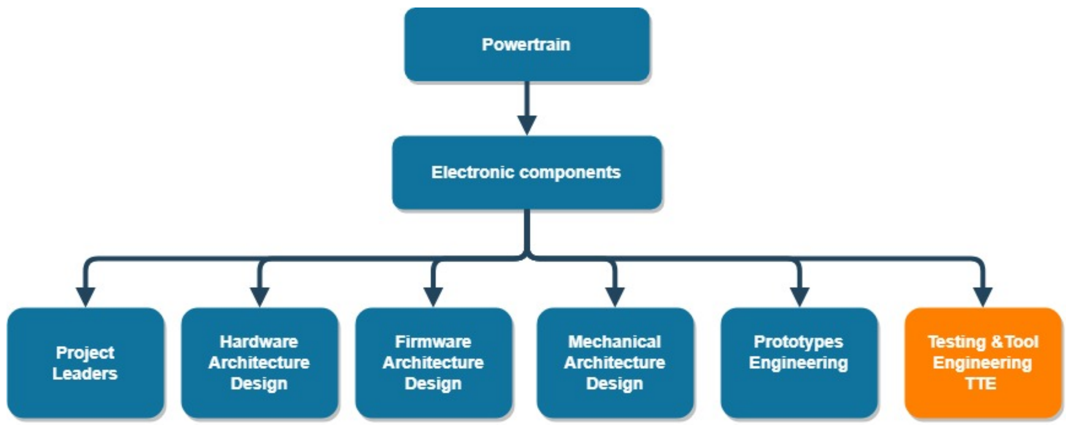

Powertrain
Electronic components
Project Leaders
Hardware Architecture Design
Firmware Architecture Design
Mechanical Architecture Design
Prototypes Engineering
Testing & Tool Engineering TTE

Figure 3.2. Magneti Marelli - PWT Department Structure

- **Project Leaders**: It is a cross-team composed of few Professional Leaders endowed with a high level know-how to be spread with other employees.
- **HW Architecture Design**: It is the team in charge of performing the whole design process of the board, from specification to layout.
- **FW Architecture Design**: It is the team in charge of developing the test FW and application FW. These will be loaded on the target ECU respectively for testing phases and for the in-field operational life.
- **Mechanical Architecture Design**: It is the team in charge of handling the mechanics of the electronic board such that connectors, seal to isolate the board, resin to isolate components and mechanical parts to be assembled to the board.
- **Prototypes Engineering**: It is the team in charge of retrieving prototypes. It is important to notice that prototypes are not produced by Magneti Marelli since the production centers work for huge amount of quantities. Thus, the production is entrusted to a supplier and an internal organization is needed to order, ship and retrieve the samples.
- **Testing & Tool Engineering** (orange highlighted in Figure 3.2): It is in charge of testing the electronic parts of all Magneti Marelli Powertrain business line products. The goal of the TTE-PWT team is to "provide to product development chain stable and reliable methodology and solution to check, validate and test final product, from the first prototype to mass production" [5].

3.2.1 Testing Team

The Testing & Tools Engineering team is structured as follows:
Four sub-teams cooperate to fulfill the TTE-PWT mission. Each team is characterized by its own missions and responsibilities:

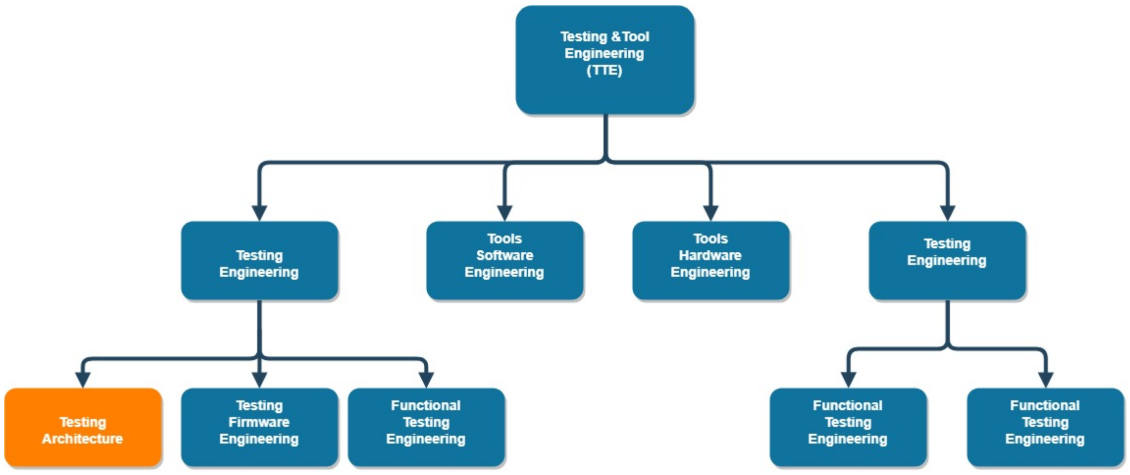

Testing &Tool Engineering (TTE)
Testing Engineering
Tools Software Engineering
Tools Hardware Engineering
Testing Engineering
Testing Architecture
Testing Firmware Engineering
Functional Testing Engineering
Functional Testing Engineering
Functional Testing Engineering

Figure 3.3. MM-PWT TTE Team Structure

- **Testing Engineering**: It involves the figures in charge of defining methodologies for the functional EOL test, developing EOL Test Program (TPGM) both for prototyping and industrialization phases, releasing coverage analysis specification and finally managing and supplying the test equipment on submitting periodical maintenance, calibration and fixture/harness design.
- **Prototyping Engineering**: It involves the figures in charge of applying functional TPGM sequences (developed by the test Engineers) to a certain number of prototype samples such that verifying the reliability of the product before starting the industrialization phase (mass production).
- **Testing SW Engineering**: It involves the figures in charge of verifying that, after production, the product passes ICT and EOL tests. Then they prepare the board for in-field operational life by replacing the test FW with the application FW and writing into the Electrically Erasable Programmable Read-Only Memory (EEPROM) the needed calibrations.
- **Testing HW Engineering**: It involves the figures in charge of applying stress tests to the products such as Power Run-In (PRI) in which some areas of the board are over-stressed to verify how certain components react. The test HW Engineer is also responsible for finding the source of errors when a fail product is sent back to the company. These technique are applied to both prototyping and returned-from-field samples [5].

3.2.2 Test Engineer Interactions

The test Engineer needs to interact with the other PWT teams in order to retrieve and share informations. These relationships involve:

- **TTE Manager**: To be aligned with shared planned activities.
- **Testing Engineering Professional Leader**: To be aligned with shared planned activities and receive technical advices related to choices to be taken during the workflow.
- **HW Designer**: To receive documents (such as schematic and layouts) needed to perform studies and analysis on the board before starting the test phases. Understanding how a product works is crucial for a Test Engineer in order to exercise proper functional tests on it.
- **FW Designer**: To receive the test firmware needed to be flashed on the board before applying the functional test.
- **Mechanical Designer**: To receive specifications relating to the output interfaces of the board (sockets for DUT - Automatic Test Equipment (ATE) connection). As soon as a new product sample arrives, it is important to have everything ready to start the test phase as soon as possible.
- **Prototype Engineer**: To receive the prototype samples which have to be tested.

Chapter 4

Battery Electric Vehicle

Battery Electric Vehicles (BEVs) and Hybrid Electric Vehicles (HEVs), in substitution of Internal Combustion Engine (ICE) vehicles, are a part of the solution to problems such as urban air pollution, fossil fuel depletion and global warming. The car market is shifting more and more towards this direction and therefore it is necessary to deepen and study more in detail the use of new technologies for the construction of electric vehicles.

4.1 BEV Powertrain

In a BEV the propulsion is based exclusively on the electric energy stored in the High Voltage (HV) battery. BEVs derive all power from battery packs and thus they have no internal combustion engine, no fuel cell, nor fuel tank. BEVs are becoming more and more attractive with the advancement of new battery technology (Lithium Ion) that have higher power and energy density. The nominal voltage is, in most of the cases, between 360V and 450V. A BEV has also a Low Voltage (LV) battery, the usual 12V battery, which is used as a power supply for the auxiliary equipment (lightning, multimedia, etc.).

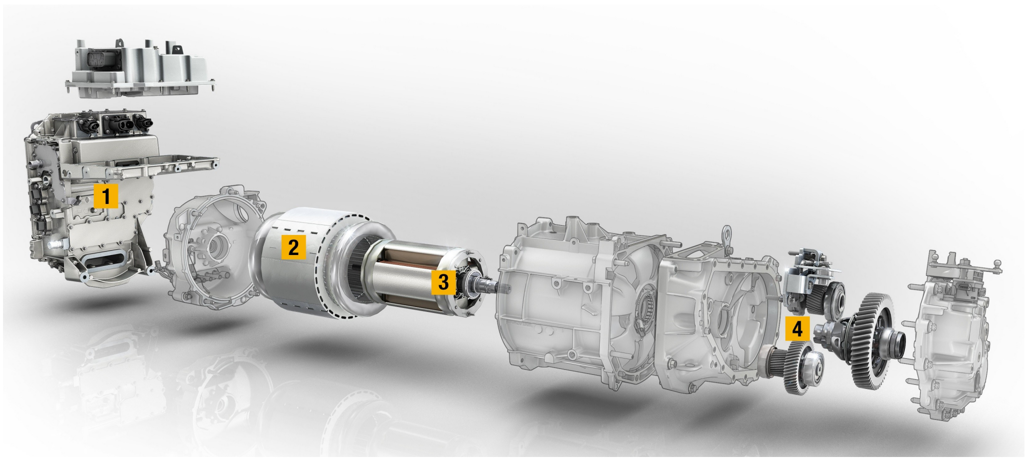


Figure 4.1. Electric vehicle powertrain

The Figure 4.1 shows a PWT of a BEV. It is slightly different from an ICE powertrain and, under certain aspects, it is simpler because it has fewer moving parts. It is composed of the following main elements:

1. Power electronic controller
2. Stator
3. Rotor
4. Single speed gearbox and differential.

When the vehicle accelerates, the electric machine takes electrical energy from the HV battery and produces torque (*motor phase*).
When the vehicle is braking, the kinetic energy of the vehicle is used by the electric machine to produce electrical energy (*generator phase*).

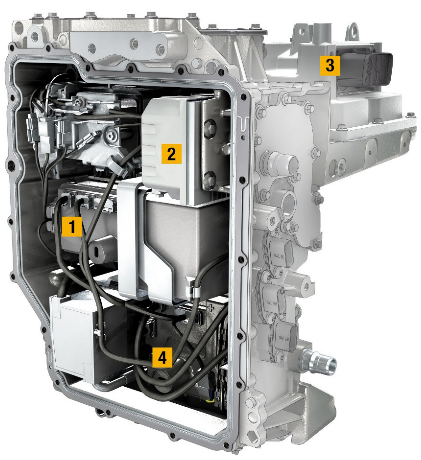


Figure 4.2. Electric vehicle power electronics

The Figure 4.2 shows the power electronics that controls an electric vehicle. It is composed of the following main elements:

1. Rectifier
2. DC-DC converter
3. Input filter
4. Inverter.

The power electronics control module has several subsystems, each responsible for a control function.

When the vehicle is charged from a home electrical grid (e.g. 220V), the **rectifier** converts the Alternating Current (AC) into Direct Current (DC), which is fed into the HV battery.

The **DC-DC converter** is responsible for the lowering of the High Voltage (e.g. 400V) to the Low Voltage network (e.g. 12V).

The **inverter** controls the electric machine speed and torque by converting the Direct Current (DC) from the battery into alternating 3-phase current for the electric machine. When the vehicle is in energy recuperation phase (*braking*) the inverter is doing the opposite conversion, from 3-phase Alternating Current (AC) to DC [6].

4.2 Inverter

Power inverters and converters are used to invert HV battery pack DC to AC for motors that propel the vehicle down the road; they also convert AC to DC to charge the HV battery pack.
In a vehicle with an electric drivetrain the engine is controlled by the inverter. The inverter determines the vehicle driving behavior exactly as it happens for gas or diesel ICE vehicles. It also captures kinetic energy released through regenerative braking and feeds this back to the battery. As a result, the range of the vehicle is directly related to the efficiency of the main inverter.

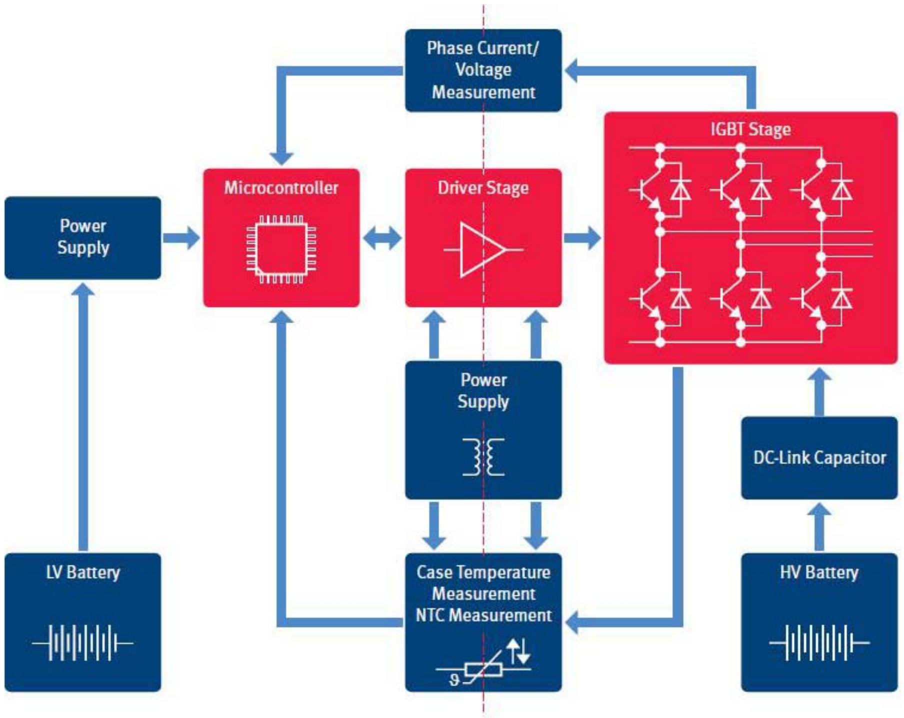


Figure 4.3. Block diagram of a main inverter for electric vehicle

The Insulated Gate Bipolar Transistors (IGBTs) are a high-voltage, high-current switches directly connected to the traction motor in HEVs or Electric Vehicles (EVs). A more efficient IGBT leads to less power lost in the form of heat, resulting in better mileage (sometimes called miles per watt of energy).

Unlike IGBTs, MOSFETs have no "tail" current when turned OFF. SiC Metal-Oxide Semiconductor Field-Effect Transistors (MOSFETs), in particular, combine several desirable characteristics, such as high breakdown voltage, low ON-resistance and fast switching speed, with their inherent advantages of high-temperature capability, high-power density and high efficiency. Moreover, their lightweight and small volume favorably affect the whole powertrain system in an HEV and, thus, the performance and cost. As a result, it is expected they will begin replacing IGBTs in HEV/EV applications [7].

4.3 DC-DC Converter

Different voltage levels are required by the various electronic components in a car or truck. The most basic requirement for DC/DC conversion is to power the traditional 12V loads. When a standard combustion engine vehicle is operating, an alternator connected to the engine provides the power for all electrical loads and also recharges the battery. The internal combustion engine in HEVs can be OFF for extended periods of time, so an alternator cannot be used to provide power to auxiliary loads. A DC/DC converter charges the 12V battery from the HV bus, thus eliminating the 14V alternator.

4.4 Battery Management System

Very large battery packs are required to power electric motors. They are usually made of hundreds of cells connected together so as to produce high voltages (400V) and high currents. Battery packs are managed and monitored by a BMS and charged via an on-board AC/DC converter module, with voltages ranging from 110V single-phase to 380V three-phase systems.

The BMS is a key element in the overall HEV and EV architecture. A good BMS can extend the battery's lifetime and it can also extend the range of the vehicle. The State of Health (SOH)[1], State of Charge (SOC)[2] and Depth of Discharge (DOD)[3] of the battery are constantly checked. A good algorithm constantly monitors these parameters and manages to modify the behavior in order to have the best performance with the least possible losses. An aspect to be taken into consideration is that the individual cells are components subject to wear. With aging the characteristics of the cells change and this negatively impacts the total battery capacity.
Fortunately, cell supervision circuitry enables cell balancing during charging and discharging phases. The vehicle power system sees the battery pack as a single

[1]**SOH**: It's a figure of merit of the condition of a battery (or a cell, or a battery pack), compared to its ideal conditions.

[2]**SOC**: It's the equivalent of a fuel gauge for the battery pack in a Battery Electric Vehicle.

[3]**DOD**: It indicates the percentage of the battery that has been discharged relative to the overall capacity of the battery.

high-voltage source, but the battery control system is able to consider each battery's condition independently. If a single cell in a stack has slightly less capacity than the other cells, then its SOC will gradually deviate from the rest of the batteries over multiple charge and discharge cycles. The more cells in series there are in a battery pack, the greater is the possible difference in SOC, impedance and capacity affecting the energy delivery of the pack.

An on-board battery management and protection system control battery state during charging and discharging to enable the longest possible battery life. Battery monitoring devices integrate all necessary components for voltage and current measurement, signal isolation and safety monitoring. Nowadays most EV and HEV battery packs are Li-ion formulations so battery protection and monitoring are mandatory. At the higher operating voltages experienced in electric vehicles, overvoltage can be catastrophic.

HEVs and EVs must be recharged on a periodic basis since batteries have a finite energy. The charging phase can be done by simply connecting the onboard charger unit to the power grid. For most users, 120V AC at 15A to 20A will be the most readily available power supply that all onboard chargers should be capable of handling. But since charging time is an important factor for car drivers, some users can take advantage of 240V AC that will allow for faster charging times, but will require a more robust power source.

The onboard charger converts electrical power from AC to DC and controls the power flow to the high voltage battery. The charging system consists of an AC/DC rectifier to generate a DC voltage from the AC line, followed by a DC/DC converter to generate the DC voltage required by the battery pack [6].

4.4.1 BMS Controller Functions

Main Battery Management System controller functions are:

- SW routines executed within protected ASIL C/D HW environment
- Main-Battery-Switch handling
- State of Health for battery monitoring subsystem
- State of Charge monitoring
- Calculation of battery cell specific model in order to increase battery lifetime over charge/discharge cycles
- International Organization for Standardization (ISO) insulation measurement
- Temperature management of battery pack (*control fan/air or pump/water*)
- Security (*battery pack manipulation protection*) [8]

Chapter 5

ECU

The Electronic Control Unit (ECU) represents the brain of the engine control and it is able to acquire the main information coming from the sensors on board and to pilot the various actuators. There are different types of ECUs depending on the specific needs of the manufacturers.

First of all you need to define the type of engine you want to analyze. There are substantial differences between motorbike engine control systems, and engine control systems for petrol and diesel cars.

In order to allow rapid times for the optimal adjustment of the control system, all the techniques and all the principles of modularity and scalability of the ECU and software have been widely used. This means that the "logical core" of the ECU remains substantially unchanged for all engines and vehicles while specific solutions are adapted and customized for each engine family or vehicle model. This also brings another significant advantage which consists in maintaining all the performances and characteristics required by the manufacturer and by the final customer, reducing costs to the only contents actually used.

The objective of the Magneti Marelli powertrain business line is to made ECUs as "platform-independent" as possible in order to adapt and reuse working parts for different customers with the lowest possible cost/risk level.

5.1 In-Vehicle Networking

In-Vehicle networking systems have evolved over the years. Present day vehicles are built with several bus systems. ECUs inside the car exchange messages between each other. For example, an ECU which contains sensors needs to send its data to another ECU responsible for controlling an actuator. Each bus system is used for a specific domain or purpose. These features could be execution of time-critical tasks, exchange of control signals or more recently the high-bandwidth data like audio/video signals from high resolution cameras used in Advanced Driver Assistance Systems (ADAS)[1] sensor fusion. The most common bus systems currently used in a vehicle are the following:

- **CAN**: The CAN, introduced in a vehicle series in 1988, is the oldest bus system and still in use in every car. Its prevalence is due to the major requirement for robust ECU communication. On the other hand, CAN networks carry the disadvantage of low bandwidth rate, restricted to 1 Mbit/s. A single CAN frame can carry a maximum payload of 8 bytes. These two factors results in its inability to transmit large data payloads that are required to implement the latest advances in ADAS domain.

- **Flexray**: Development for Flexray was started in the early 2000s with the major goal of replacing the conventional mechanical functions by electrical functions, with the so called X-bywire (e.g Brake-by-Wire or Steer-by-Wire). High requirements with respect to safety in these application fields also led to the introduction of this new bus protocol. Another improvement is the bandwidth rate (upto 10Mbit/s) compared to CAN. This protocol is implemented with Time Division Multiple Access (TDMA) principle, compared to the event triggered Carrier Sense Multiple Access/Collision Resolution (CSMA/CR) for CAN. Flexray also incorporates event based triggering with the use of dynamic slots in its frame format. Static slots are followed by dynamic slots.

- **LIN**: The Local Interconnection Network (LIN) is a one-wire bus. It is a cheaper alternative to CAN and is mainly used in smaller environments e.g. for interconnecting all sensors and actuators of a door or a seat (sub-systems). The main consideration for LIN was the cost factor and simplicity in its design, thus the transmission rate was less important at the time of its consideration.

- **MOST**: The Media Oriented Systems Transport (MOST) has been designed for multimedia applications in 1998. Upto 64 ECUs can be connected with each other using a ring topology. The recent MOST150 standard supports transmission rates up to 150 Mbit/s. Nearly all the major car manufacturers use MOST inside their cars for the infotainment and telematics applications. Compared to CAN and Flexray, there are far less ECUs inside the vehicles using the MOST bus network. MOST is a proprietary technology invented by SMSC (now taken over by MICROCHIP) and currently maintained by the MOST cooperation (AUDI, BMW, SMSC, Harman & Becker). For this reason, it is a very expensive technology and also prevents MOST from being used for in-vehicle communication outside the infotainment network, irrespective of its high bandwidth capability.

[1]**ADAS**: Aim to drastically reduce road accidents and the associated casualties by helping drivers avoid collisions altogether. These systems react faster than any human, they are constantly vigilant, and they are already being adopted and deployed across various car segments, from premium to economy models.

- **Ethernet**: Ethernet is the newest bus system introduced into the in-vehicle networking domain. Ethernet, though has been a solid communication network over the years as a comprehensive office and home network. It was a never choice due to its expensive cabling and strict Electromagnetic Compatibility (EMC)[2] requirements for the automotive domain. Now thanks to increasing demands for higher bandwidth, and with limitations of existing bus systems with regards to in-vehicle gateway networks, suddenly Automotive Ethernet has gained a lot of footholds. It was first introduced for diagnostics of ECUs with external tools and for ECU ashing purposes due to its high bandwidth. During the second generation (current phase), it has gained more presence in the ADAS domain as well as into in-car networking between ECUs. It is predicted that in the coming years, Automotive Ethernet becomes the de facto standard for in-vehicle networking whereby all the several application domains such as Powertrain, Chassis, Multimedia, ADAS are interconnected via Ethernet cable [9][10].

5.2 Test

For the classic internal combustion engine everything turns around the test of physical parameters. The keywords are combustion dynamics, pressure, temperature, fluids, mechanical couplings and power transmission, emissions management, as well as turbos, superchargers and all methods to increase the efficiency or power of controlled explosions. The goal is to transform the linear movement of the piston into a rotary motion and balance the output power with the flywheels. For the electric powertrain, everything turns around the electrical test. The key parameters are related to power electronics:

- Switching frequencies
- Voltages and currents
- Induction and counter-Electromotive Force (EMF)
- Battery capacity and discharge speed
- Thermal management of inverters and converters
- Regulation of power regeneration
- Engine/generator phase angles and sheet geometries
- Magnet position
- Flow lines.

[2]Electromotive force: it's the branch of electrical engineering concerned with the unintentional generation, propagation and reception of electromagnetic energy which may cause unwanted effects such as Electromagnetic Interference or even physical damage in operational equipment.

Electric motors and inverters respond more quickly than thermal motors and have a decidedly non-linear behavior in their field of operation.
The control signals from the ECUs are very fast (2-20 kHz) so the specialized models that simulate the behavior of the motors must work 100 times faster (from 200 kHz to 2 MHz) to represent them precisely in a system testing with HIL techniques.
This cannot be achieved efficiently on systems based on the normal processors that are used for HIL tests on thermal motors ECUs. Thus, suppliers of measurement and test systems such as National Instruments are developing simulation tools based on FPGAs, to facilitate the creation of models from specific electrical modeling tools needed to work with control loop speeds of the order of microseconds.

5.2.1 Power-level test

Usually the ECU and the inverters are assembled together. This makes testing at the signal level slightly difficult (from -10 to 10 V and a few mA). It is more appropriate to perform the test at full power, delivering or absorbing the actual current levels of the motors, rather than opening the ECU to perform the tests.
However, this means testing at power levels up to 200 kilowatts. These power levels require measurement equipment with isolated channels and a power supply system capable of dynamically delivering or absorbing load currents of this magnitude.

5.2.2 Battery pack

Batteries, especially those designed for larger capacity plug-in hybrid vehicles, must be characterized at the level of single cell, module and group.
Performing this type of test can be complex (or have prohibitive costs) due to the range of voltages (0V-800V) present on the test points of the battery pack with cells configured in series/parallel and due to the precision required for measuring the voltage of common way.

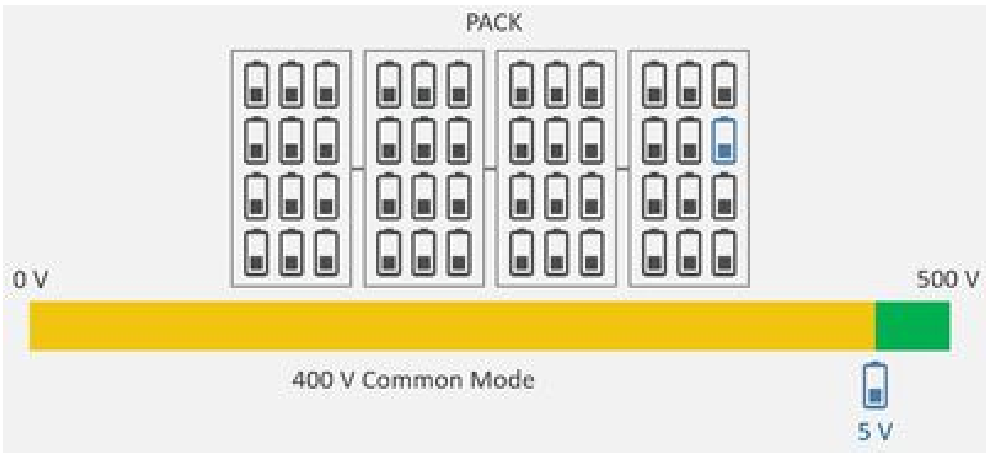


Figure 5.1. Battery pack

The Figure 5.1 show the common architecture of a battery pack (cell, module, group) and range of measurement of the necessary voltage.

The group is essentially equipped with its own electronic control unit, the Battery Management System (BMS). This is tested at subsystem level and at component level using a battery simulator in order to check algorithms and control functions.
These tests are carried out in thermal chambers (battery behaviour depends on the temperature) and include both characterization and duration tests. A key attribute of the performance of a battery pack is the behavior of the charge/discharge cycle during of its useful life and in a wide range of operating temperatures (How long will the battery last under normal conditions of use in different climates?).
To obtain test results within acceptable time limits and with statistical relevance, many car manufacturers carry out numerous (from tens to several hundred) tests of parallel groups. Managing this volume of testers and generated data, ensuring the traceability of results and the security of the validity of the tests, requires tools for the automation of test procedures, system-level management and the presence of systems for the management of data designed specifically for these use cases.

Chapter 6

AUTOSAR

6.1 History

The AUTOSAR development partnership was formed in July 2003 by BMW, Bosch, Continental, Daimler, Chrysler, Siemens VDO and Volkswagen to develop and establish an open industry standard for automotive E/E architecture. In November 2003, Ford Motor Company joined as a Core Partner, and in December Peugeot Citroën Automobiles S.A. and Toyota Motor Corporation joined. The following November General Motors also became a Core Partner. After Siemens VDO was acquired by Continental in February 2008, it ceased being a self-contained Core Partner of AUTOSAR (Figure 6.1).

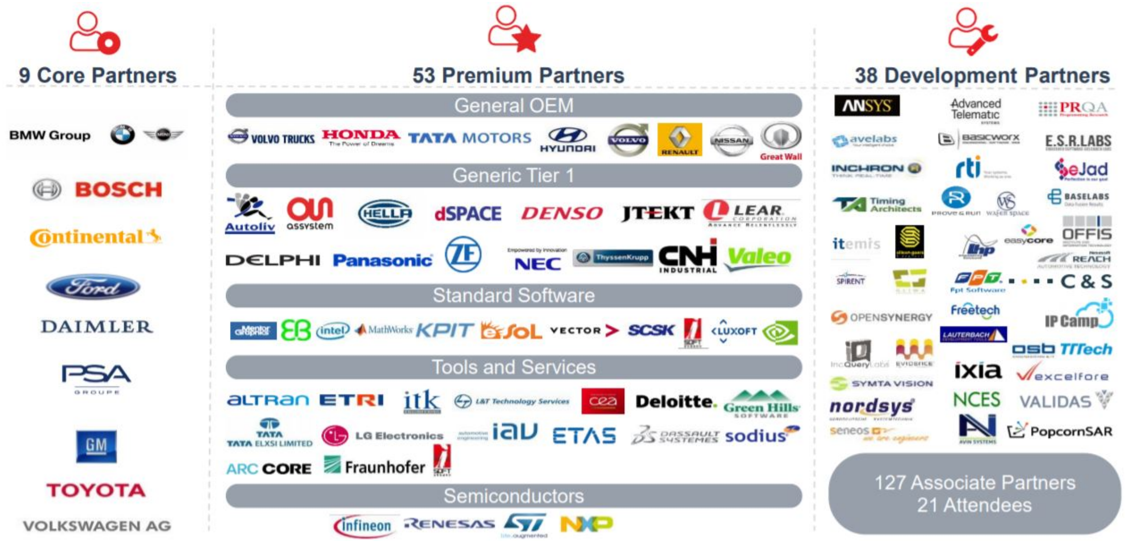


Figure 6.1. AUTOSAR core partners and partners

Since 2003, AUTOSAR has provided four major releases of the standardized automotive software architecture for its Classic Platform and one release of Acceptance Tests.

The work of AUTOSAR can be divided into three phases:

1. (2004-2006): Basic development of the standard (Releases 1.0, 2.0 and 2.1).
2. (2007-2009): Extension of the standard in terms of architecture and methodology (Releases 3.0, 3.1 and 4.0).
3. (2010-2013): Maintenance and selected improvements (Releases 3.2, 4.1 and 4.2).

In 2013, the AUTOSAR consortium entered a continuous working mode for its Classic Platform to maintain the standard and provide selected improvements (including the release R4.2 as well as Release 1.0 of Acceptance Tests).
In 2016 the work on the Adaptive Platform started. A first release (17-03) was published in early 2017, followed by release 17-10 in October 2017 and release 18-03 in March 2018. The goal is to conclude the major development activities in a joint release of AUTOSAR Classic, Adaptive and Foundation in October 2018 [11].

6.2 Generalities

AUTOSAR is an open and standardized automotive software architecture, jointly developed by automobile manufacturers, suppliers and tool developers. The AUTOSAR-standard enables the use of a component based software design model (Figure 6.2) for the design of a vehicular system. The design model uses application software components which are linked through an abstract component, named Virtual Functional Bus (VFB).

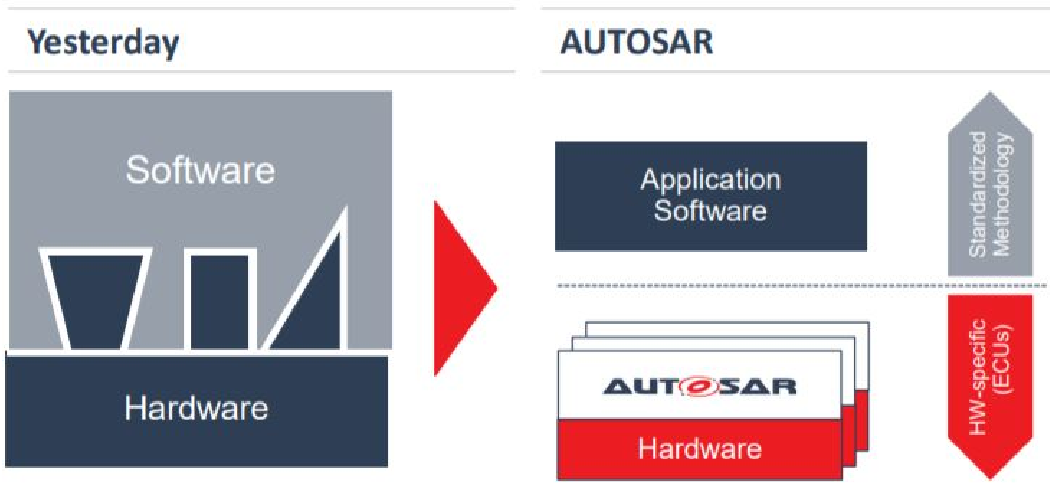


Figure 6.2. AUTOSAR modularity - layered architecture

The *AUTOSAR Application Software Components (SWCs)* are the smallest pieces of application software that still have a certain functionality. The software of an application can then be composed by several SWCs. Standardized interfaces to build different automotive applications are specified in the AUTOSAR-standards.

The *VFB* connects the different SWCs in the design model. This abstract component interconnects the different application SWCs and handles the information exchanged between them. The VFB is the conceptualization of all HW and system services offered by the vehicular system. This makes it possible for the designers to focus on the application instead of the infrastructure SW.

Thanks to the VFB a SWCs do not need to know with which other SWCs need to communicate. Each SWCs give their output to the VFB, which guides the information to the input ports of the SWCs that need that information. This is possible due to the standardized interfaces of the SWCs which specifies the input and output ports as well as the format of data exchange.
This approach makes it possible to validate the interaction of all components and interfaces before SW implementation. This is also a fast way to make changes in the system design and check whether the system will still function.
The AUTOSAR-project created a methodology that can be used to create the E/E system architecture starting from the design-model.

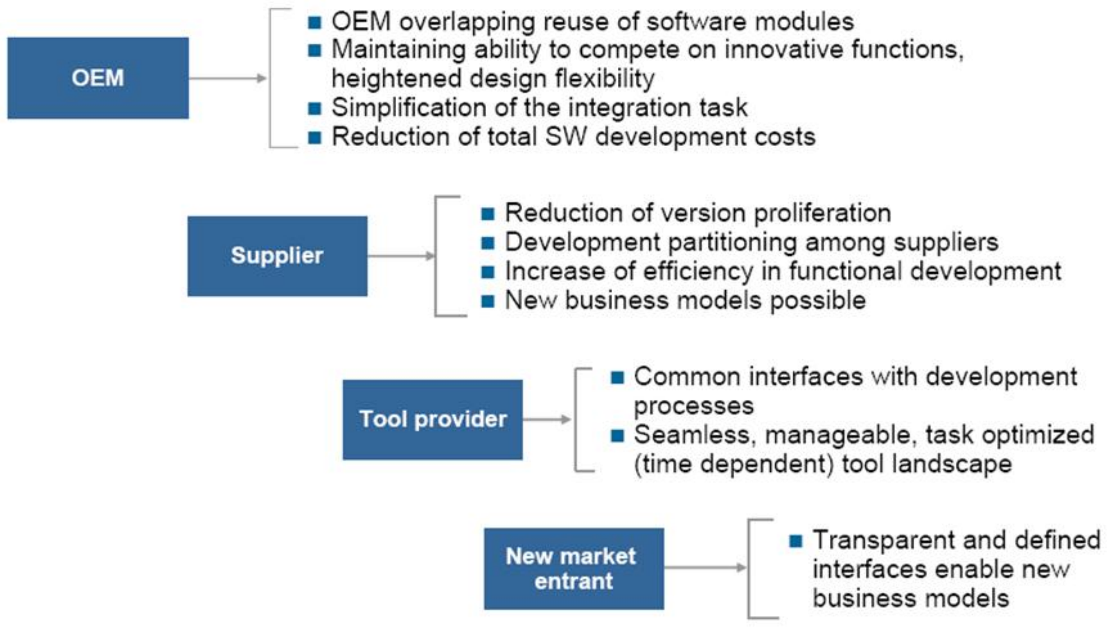

OEM
OEM overlapping reuse of software modules
Maintaining ability to compete on innovative functions, heightened design flexibility
Simplification of the integration task
Reduction of total SW development costs
Supplier
Reduction of version proliferation
Development partitioning among suppliers
Increase of efficiency in functional development
New business models possible
Tool provider
Common interfaces with development processes
Seamless, manageable, task optimized (time dependent) tool landscape
New market entrant
Transparent and defined interfaces enable new business models

Figure 6.3. Benefits of AUTOSAR for different actors in the car-makers scenario

AUTOSAR aims to improve complexity management of integrated E/E architectures through increased reuse and exchangeability of SW modules between Original Equipment Manufacturers (OEMs) and suppliers.
The goal is to standardize the Software architecture of an ECUs.
As shown in Figure 6.3, the AUTOSAR standard has the following features:

- Hardware and software widely independent of each other.
- Development can be decoupled by horizontal layers, reducing development time and costs.
- Reuse of software enhances quality and efficiency [12].

6.3 AR Layered Architecture

The AUTOSAR layered architecture is offering all the mechanisms needed for software and hardware independence. It is composed of three main software layers which run on a Microcontroller (µC):

- Application Layer
- Runtime Environment (RTE)
- Basic Software (BSW).

The Basic Software is further divided into the following layers (Figure 6.4):

- Services Layer
- ECU Abstraction Layer
- Microcontroller Abstraction Layer (MCAL)
- Complex Drivers [12][13].

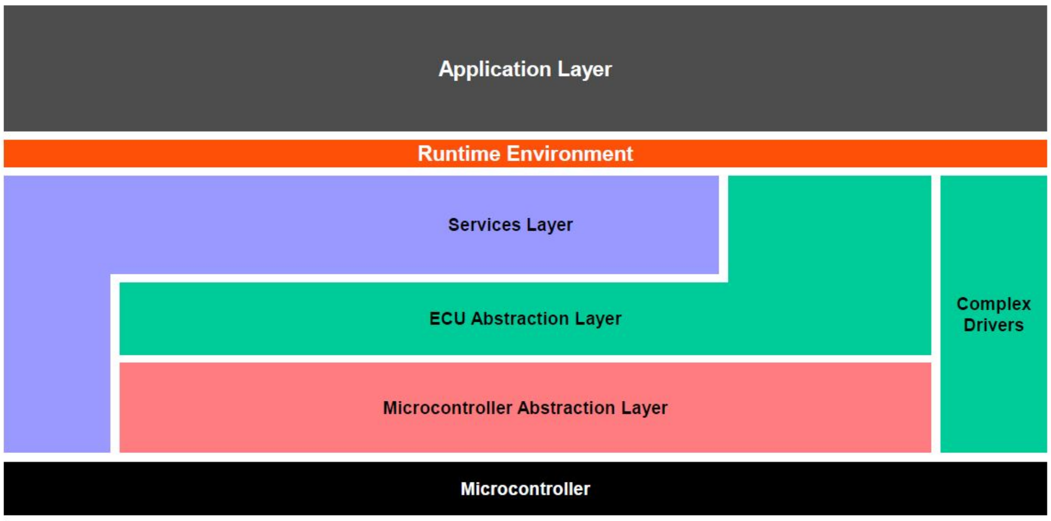


Figure 6.4. AUTOSAR layered architecture

6.3.1 Microcontroller Layer

This is the lowest layer in the architecture. It is the hardware on which the entire AUTOSAR architectural layers are built upon. It contains the input/output peripherals, communication peripherals, the memory interfaces and microcontroller chip itself that make up the ECU [12][13].

6.3.2 Basic Software Layer

Microcontroller Abstraction Layer

The Microcontroller Abstraction Layer (MCAL) makes upper software layer independent of Microcontroller Unit (MCU). It is a collection of software modules that directly access on-chip MCU peripheral modules and external devices that are mapped to memory.
MCAL enables a very significant advantage of the layered architecture of the AUTOSAR compliant design; it makes the application and also the middleware (BSW layer) independent of the underlying hardware platform.
Figure 6.5 shows in detail the MCAL software module architecture, and the different on-chip peripheral functions available on a microcontroller.

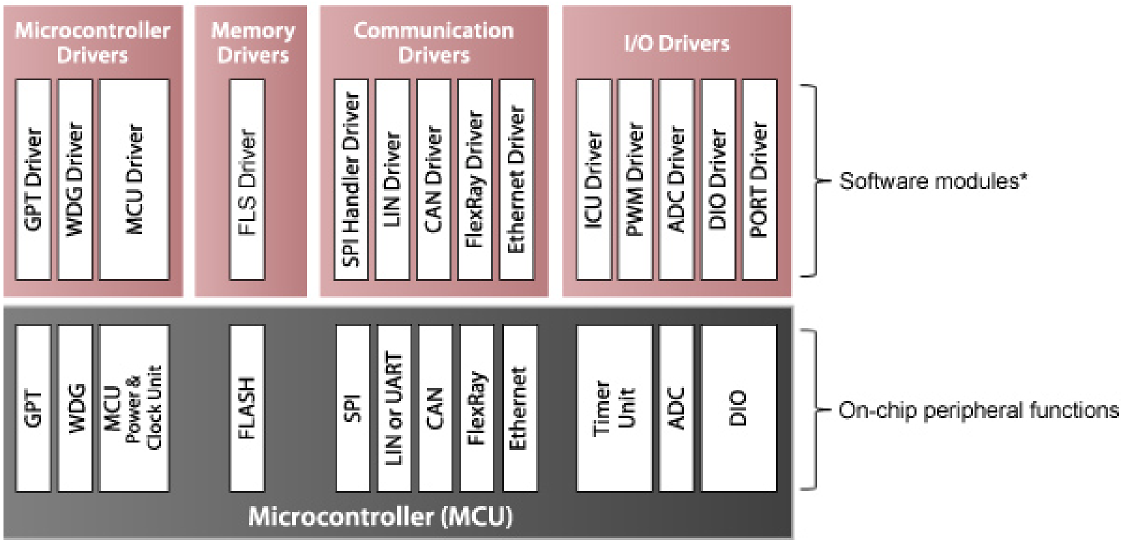


Figure 6.5. AUTOSAR MCAL software module architecture

ECU Abstraction Layer

It makes upper software layer independent of ECU hardware layout. It interfaces with MCAL (including external device driver) to provide:

- Access to peripherals and devices irrespective of whether they are inside or outside the MCU.
- Application Programming Interface (API) for interfacing with MCU (port pins, interface type) (General purpose I/O, ADC, Serial Peripheral Interface (SPI), PWM).

Microcontroller Drivers

- GPT Driver: General Purpose Timer (GPT) device driver uses on-chip MCU timer. Initializes GPT and performs timer count.

- WDG Driver: Watchdog (WDG) Driver, this on-chip device driver Initializes WDG and performs WDG mode settings.
- MCU Driver: Microcontroller Unit (MCU) Driver, this device driver helps configure MCU settings, initializes clock and helps configure power mode settings.

Memory Drivers

- FLS Driver: Flash (FLS) Driver initializes FLS and reads/writes to FLS memory.

Communication Drivers

- SPI Handler/Driver: Serial Peripheral Interface (SPI) is a Handler/Driver Device with on-chip clock serial function that Initializes SPI, performs SPI I/O and SPI I/O buffer settings.
- LIN Driver: LIN is a device driver that initializes LIN and performs LIN I/O.
- CAN Driver: CAN is a device driver that initializes CAN and performs CAN I/O.
- FlexRay Driver: FlexRay device driver initializes FlexRay and performs FlexRay I/O.
- Ethernet Driver: Ethernet device driver initializes Ethernet Driver and performs Ethernet Driver I/O.

I/O Drivers

- ICU Driver: Input Capture Unit (ICU) is a device driver using on-chip MCU timer and initializes Input Capture Unit (ICU). It also measures PWM waveforms.
- PWM Driver: PWM is a device driver using on-chip MCU timer. It initializes PWM and sends PWM waveforms as output.
- ADC Driver: Analog Digital Converter (ADC) is a device driver for on-chip Analog Digital Converter (ADC). It Initializes ADC, starts/stops AD conversion, sets AD conversion result buffer and reads AD conversion results.
- DIO Driver: Digital Input/Output (DIO) is an MCU port device driver that performs port signal (input/output).

Service Layer

The Services Layer is the highest layer of the Basic Software which also applies for its relevance for the application software: while access to I/O signals is covered by the ECU Abstraction Layer, the Services Layer offers:

- Operating System (OS) functionality
- Vehicle network communications and management services.
- Memory service (NVRAM management).
- Diagnostic service (including UDS communication, error memory and fault treatment).
- ECU state management.
- Mode management.
- Logical and temporal program flow monitoring (Watchdog manager).

Complex Drivers

A Complex Driver is a software entity not standardized by AUTOSAR that can access or be accessed via AUTOSAR Interfaces and/or basic software APIs.
It implement complex sensor evaluation and actuator control with direct access to the microcontroller using specific interrupts and/or complex microcontroller peripherals, external devices (communication transceivers, ASIC...) to fulfill the special functional and timing requirements [12][13].

6.3.3 Run Time Environment

It provides communication services for the application software. It makes SWCs independent of the mapping to specific ECUs. It is generated automatically from the SWCs mapping on the ECUs [12][13].

6.3.4 Application Layer

It is where the AUTOSAR application is defined. The application is defined as a set of SWCs that communicate with other SWCs and/or services via the RTE. Each SWC is defined through a port interface (how the SWC communicate with other SWC) and an internal behavior (*task*) [12][13].

6.4 AR Vehicle Diagnostics

In automotive, diagnostics is required to be performed on all ECUs to ensure there is no issue with any electronically controlled component of the vehicle. Any issue encountered by the automotive ECU is stored as Diagnostic Trouble Code (DTC) in the EEPROM. These codes can be retrieved later using an automotive diagnostic tool. The vehicle diagnostics system needs to be implemented in the AUTOSAR architecture.

6.4.1 Diagnostics Communication Manager

In the AUTOSAR Architecture the DCM is located in the Communication Services (Service Layer). The DCM has the responsibility of reading and writing the fault codes or diagnostic trouble code in the fault memory of the automotive ECU. It supports the implementation of the diagnostic protocols such as UDS (ISO 14229) and OBD II.
When an automotive ECU receives a diagnostics request from the tester tool, the DCM pre-processes it. While it handles majority of the requests, any other request is routed to the functional software. Every new version of DCM has an enhanced functional range which increases its ability to handle different types of diagnostic requests.
DCM comprises of the Service Identifiers (SID), Data Identifiers (DID) sub-functions and routine identifiers to handle the vehicle diagnostics requests.
While handling the diagnostics requests from external testing tools during ECU software development, vehicle program production or garage servicing, the DCM also manages the session and security states.
For instance, the DCM checks whether a particular diagnostic request can be supported and also if the execution of that service will be carried out in the current session.
We will deliver little deeper into the different components of DCM and how they interact with each other.

- Diagnostic Session Layer: This sub-module is tasked with ensuring the flow of data related to the diagnostic requests and the responses. Diagnostic session and security states are managed by this layer along with the supervision of timing parameters of the diagnostic protocol.
- Diagnostic Service Dispatcher: The validity of the incoming diagnostic requests needs to be checked and this is where this sub-module comes into the picture. In addition to ensuring the validity of the request, it also keeps track of the progress of the request. After the validation is done, this sub-module processes a stream of diagnostic data and also transmits the response post-data processing.
- Diagnostic Service Processing (DSP): This module analyzes the format of the service request and interacts with other components of the BSW to collect the data required for the processing. It also assembles the service response.

These three sub-modules interact with each other through a well-defined interface. In order to make the automotive ECU an AUTOSAR compliant product, DCM needs to be first configured using tools like Vector DaVinci. During the development process, all the necessary APIs are coded in the DCM including the general DCM definitions. Unit testing and integration testing follows the development process.

Unit testing is performed by using tools like Tessy Test Systems where the source module is analyzed and the functions are tested.

Integration test is performed on the evaluation board where the data exchanged between the modules are verified. Basically, it tests if all the interfaces are correctly working without any issue [14].

6.4.2 Diagnostics Event Manager

The Diagnostics Event Manager (DEM) is responsible for storing and processing diagnostic errors (*events*) and all the data associated with it. In addition to it, DEM provides the DTC to the DCM as and when required.
Providing the interface to the application layer of the ECU and to other modules of the AUTOSAR Base Software Module is also one of the responsibilities of DEM.
There can be two type of event that can be reported to DEM:

- Base Software Event: Event reported by the base software.
- Software Component Event: reported by the AUTOSAR application layer.

The events reported to the DEM need to be first qualified to ensure if it is a fault (*failure of a component*) or just an occurrence (*irregular system behavior*).
After the event is qualified, DEM records the event data and communicates with DCM for event handling and Function Inhibition Module (FIM) for functional control [15].

6.4.3 Function Inhibition Module

The FIM is essentially the control mechanism for AUTOSAR Base Software and software components. The FIM has to control the functionality available to these components depending on their inhibit conditions.
An identifier is assigned to the functionalities with an inhibit condition. Only in the scenario of inhibit condition being not true, the functionality is executed.
The role of FIM is to configure and modify the inhibiting conditions of the functionalities. By doing so, a particular functionality can be adapted easily to a new system context.
FIM services are primarily focused on the applications residing in the software components; however, the AUTOSAR BSW can also use the services of FIM when required.
The FIM has a close connection with DEM as the diagnostic events and their status info are considered as inhibit function by the FIM. When a failure is detected, it is reported to the DEM and it is the job of FIM to stop the particular functionality.

Chapter 7

ECU Architecture

7.1 Original architecture

Now the layered architecture studied in Magneti Marelli S.p.A. and used originally will be analyzed. The following architecture refers to a generic ECU in the power-train area.
The original SW architecture (no longer used) is based on 3 main layers:

- Application (application strategies layer)
- Base (basic software layer)
- BIOS (microcontroller layer).

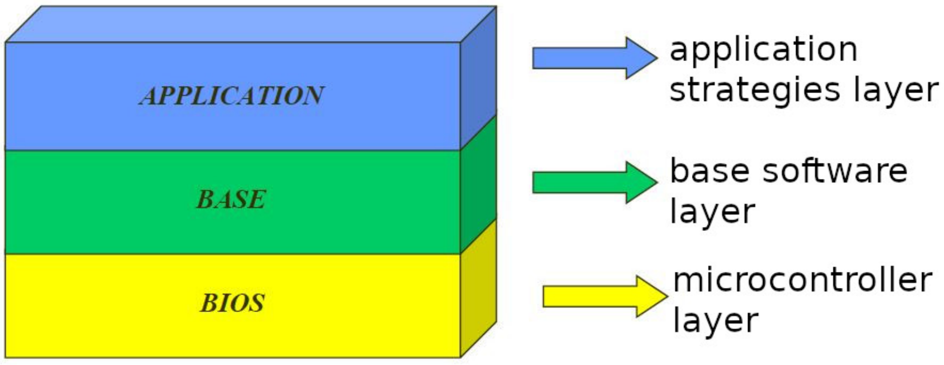


Figure 7.1. MM - ECU original architecture

The limits of this architecture are very evident. Only the abstraction from the microcontroller was carried out, and all the rest was rewritten for each different ECU; therefore, there was the need to study and migrate to a new architecture.
The study of the new architecture focused on a few points:

- Analysis of software contents in current motor control applications.

- Search for items common to different customers.
- Subdivision in software layers according to the dependencies of the system elements [16].

7.2 Current architecture

The work done by Magneti Marelli engineers during the years was to restructure the architecture of the PWT ECU with the aim to define a new software layer capable to create a "platform-independent" system. This means having a single system adaptable on different ECUs with the lowest possible cost/risk level.

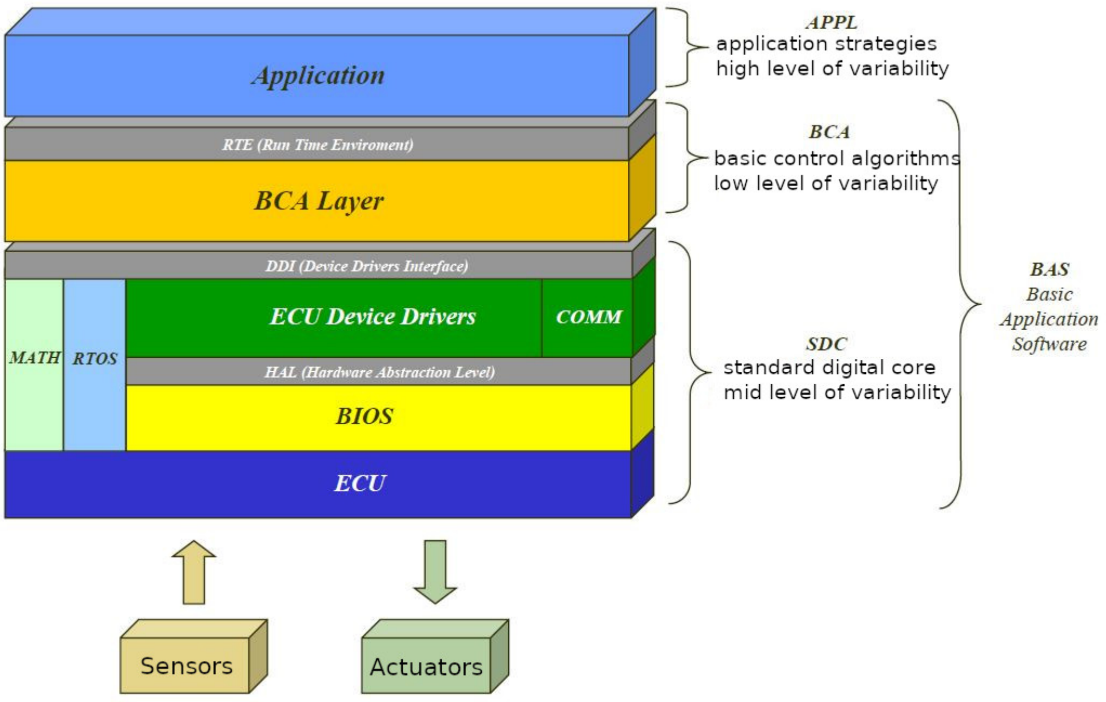


Figure 7.2. MM - ECU current architecture

The Figure 7.2 shows the result of the Magneti Marelli-PWT team study. A subdivision in software layers has been designed, in operation dependencies on system elements [16].

7.2.1 SDC

The Standard Digital Core (SDC) deals with the writing of the BIOS and ECU Device Driver (EDD) functional specifications and carry out the implementation and testing. It also provides three parallel layers related to communication (COMM), operating system (RTOS) and mathematics functions (MATH). It deals with the configuration of the layers of competence based on the characteristics of the system and hardware platform.

BIOS

It deals with dependencies from microprocessor. It is the only software level capable to access resources directly from the microcontroller.

ECU Device Drivers

It encapsulates all dependencies from the control unit hardware. Its sector of operation is the ECU connector. It treats essentially magnitudes of electrical type (ex: voltages/currents) or duration (ex: times/angles).
ECU Device Drivers main functions are:

- Acquiring analogue quantities, executing them for the electrical diagnosis, make the possible calibration hardware and scaling to compensate any signal partitions performed hardware. The data produced at Device Driver Interface (DDI) level is the voltage seen on the pin of the ECU connector.

- Acquiring slow digital signals and carry out the debouncing. The result of this operation will be the logical information of the acquired signal.

- Acquiring frequency signals by providing information on their physical characteristics like the period or the duration of an impulse.

- Managing the implementations of a frequency type (on time/angle basis) and carrying out the electrical diagnosis.

- Managing SPI communication for receiving informations transmitted by the custom (digital diagnosis and inputs), and for the transmitting informations to custom (serial implementations).

- Managing the physical devices mounted on the ECU, like the serial Erasable Programmable Read-Only Memory (EPROM) and the custom, which perform complex signal processing such as linear lambda probe and detonation [16].

7.2.2 BCA

The set of the Sensors-Actuators Layer and System Layer is called Basic Control Algorithms (BCA). The BCA layer manage the features of the sensors and actuators connected to the ECU. The BCA layer knows the set of sensors/actuators and it is able to manage the exchange of information between them.

7.2.3 APPLICATION

The Application Layer contains the system's control algorithms. It has a high level of variability due to customer requests. It is normally developed with Model Based approach and it synthesis automatic code.

7.3 New ECU architecture AUTOSAR-based

The need to migrate to an AUTOSAR system has made it necessary to study and implement a new architecture that is perfectly compatible with the AUTOSAR standard and specifications. An in-depth study and collaboration between different actors of the Magneti Marelli-PWT in the realization of an ECU was necessary in order to define an interface as standard as possible.

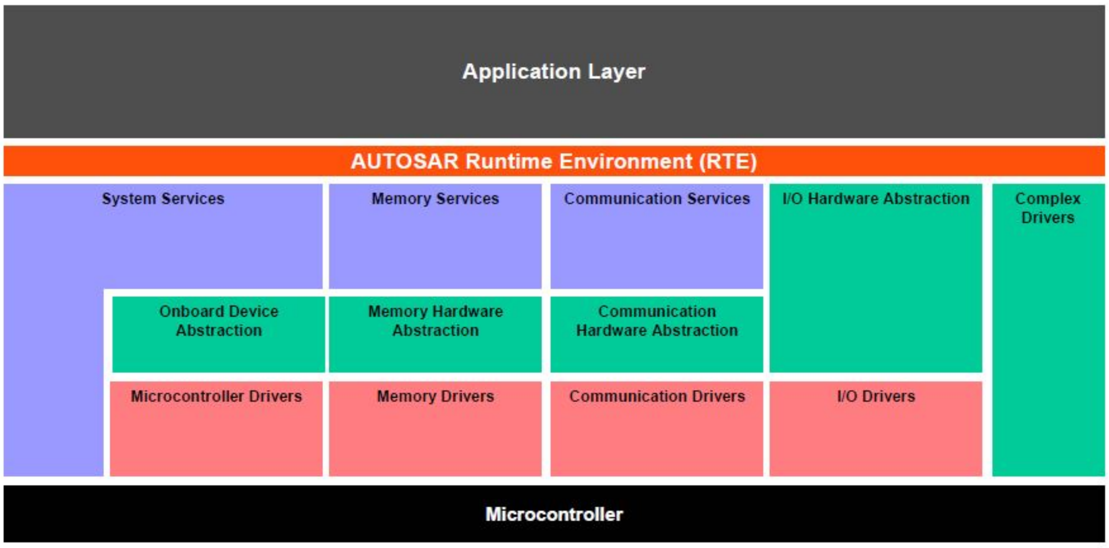


Figure 7.3. Detailed view of the AUTOSAR architecture

7.3.1 Testing

For testing purpose, the same methods and techniques used for MM-ECUs can be used for AR-ECUs even if the internal architecture is slightly different. The ECU is viewed as a black box, only the following items need to be considered:

- Network Management: AUTOSAR has defined its own Network Management (NM) protocol, which differs from the previously used one OSEK[1] NM. The test environment must take this network management protocol into account and correctly prepare and process the related messages on the network channels.

- File formats for the description of network communication: The description of network communication in AUTOSAR is part of the System Description. Previous formats such as .dbc, FIBEX or .ldf, depending on requirements of the OEM, are replaced by the new format. The test environment must be able to process this format.

[1]Open Systems and their Interfaces for the Electronics in Motor Vehicles (OSEK): It's designed to provide a standard software architecture for the various ECUs throughout a car.

As an additional value in terms of testing and debugging ECUs, AUTOSAR provides a standardized internal software architecture that ensures that certain variable sates exist in every AUTOSAR ECU and they can be used in the test environment. For example the ECU state, which is available in the EcuM module, and communication states of the individual network channels that are stored in the COMM module.

With appropriate implementation of the BSW modules, it is possible to access to these state variables over an XCP connection to the ECU (e.g. over one of the networks or a debugging interface such as JTAG or Nexus). A matching description file (A2L) for these variables is provided by the BSW generators. As an alternative, the Monitoring and Debugging protocol that was specially defined by AUTOSAR for this purpose may be used.

AUTOSAR also offers benefits in terms of access to the application level. For example, the RTE can be generated in such a way that it is possible to access the data exchanged between the SWCs. Again, a suitable A2L file can be produced by the RTE generator.

7.4 UDS Protocol

Unified Diagnostic Services (UDS) is a diagnostic communication protocol in the ECU environment within the automotive electronics, which is specified in the ISO 14229.

The diagnostic tool can contact all the control units installed in a vehicle which have an UDS services enabled. In contrary to the CAN protocol, which only uses the first and second layers of the OSI model, UDS services use the fifth and seventh layers of the OSI model. The Service ID (SID) and the parameters associated with the services are contained in the 8 data bytes of a message frame issued from the diagnostic tool.

Modern vehicles have a diagnostic interface for off-board diagnostics which makes it possible to connect a computer (client) or a diagnostics tool (tester) to the bus system of the vehicle. Thus, the messages defined in UDS can be sent to the controllers which must provide the predetermined UDS services. This makes possible to interrogate the fault memory of the individual control units or to update them with a new firmware [2].

7.5 UDS module in ECU architecture

In the layered ECU architecture there's also a communication layer (COMM) that contains an UDS module intended to serve UDS protocol functions.
The UDS module shall be interfaced to the lower layer software module and to the applicative layer. Its aim is to manage UDS services messages, forwarded by the

[2] Luca Ledda - Magneti Marelli UDS internal material 2019.

lower layer in the form of a byte stream.
In particular the UDS module shall perform the following tasks:

- Receiving request messages as primitives from a lower layer.
- Checking request messages verifying the service ID and other fields.
- Processing information to provide services managed by COMM layer and performing additional checks on received messages verifying that they are compliant to the protocol.
- Communicating with the upper layer through callbacks, to forward services managed by the applicative layer or to verify that applicative conditions to providing COMM managed services are satisfied.
- Partially writing positive responses (SID+40 byte and request message echoes) and writing all bytes of negative responses on TX buffer, calling functions defined in the lower layer [17].

7.5.1 Diagnostic Frame Format

UDS protocol works on the CAN bus protocol, so that it should not exceed the 8-bytes of data that will be used to request frame and get the response in a message.

Whenever the client wants to send a request to the ECU, it sends a request frame by filling all the fields according to some rules:

- Service ID
- Sub-Function ID
- Data Bytes.

These fields should be compliant with the available services that the ECU can provide.

In case of good UDS request is returned a positive response message followed by the service ID for the reference.

In case of bad UDS request is returned a negative response, the message coming back should contain some information about the exception occurred.

Errors have to be managed in order to let the tester know why the message response is negative. In that case the problem could be an unknown service ID, an unknown sub-function ID or some other internal exceptions. The tester always has to know many details as possible in case of failure.
UDS services allow a tester (client) to control diagnostic functions in an on-vehicle ECU (server) applied for example on electronic fuel injection, automatic gearbox, anti-lock braking system etc., connected on a serial data link embedded in a road vehicle.

This is the most typical network configuration of the client-server communication for vehicle diagnostic.

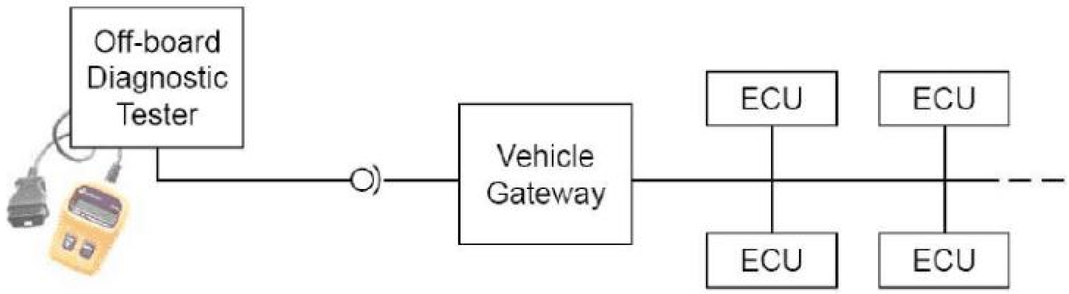


Figure 7.4. Network configuration for vehicle diagnostic

The client is usually an off-board tester. Communication is based on a request-response model. When executing diagnostic over CAN, the client initiates a request and waits for confirmation. The server receives the indication and sends a response [18].

Part II

TESTING

Chapter 8

Testing

The focus of this chapter is on the development of test automation for the automotive system model. The system model stands for the total system deemed to be a distributed system with its networked Hardware and Software.
Model Based Testing (MBT) is the application of model based design for designing and optionally also executing artifacts to perform software testing. Models can be used to represent the desired behavior of a System Under Test (SUT), or to represent testing strategies and a test environment.

8.1 State Of The Art

The basic scenario to get test cases in MBT is represented in Figure 8.1(A), where there is a test designer with its test model. The tester, expecting to have the entire test cases, executes the textual test cases. The prospective goal is to reach scenario shown in Figure 8.1(C). The goal is to find a way to write in a formal way the test cases, starting from a well-defined model. Once a well-defined format to write test cases is found, it becomes possible a test automation.
With the mentioned focus on the total system as a distributed system, the test case models become complex. Due to the many options of partitioning the variable number of participants (SWCs) to the different hardware options, and regarding the standardized architecture framework, the complexity of the models grows.
Model based testing methods are the following:

- **Unit Test** (*component test*): individual program components are tested in isolation.

- **Integration Test**: testing of the interaction of several components.

- **System Test**: to ensure the operability of the entire system according to the requirements.

- **Acceptance Test**: the test under real operating conditions, as well as the interaction of several systems.

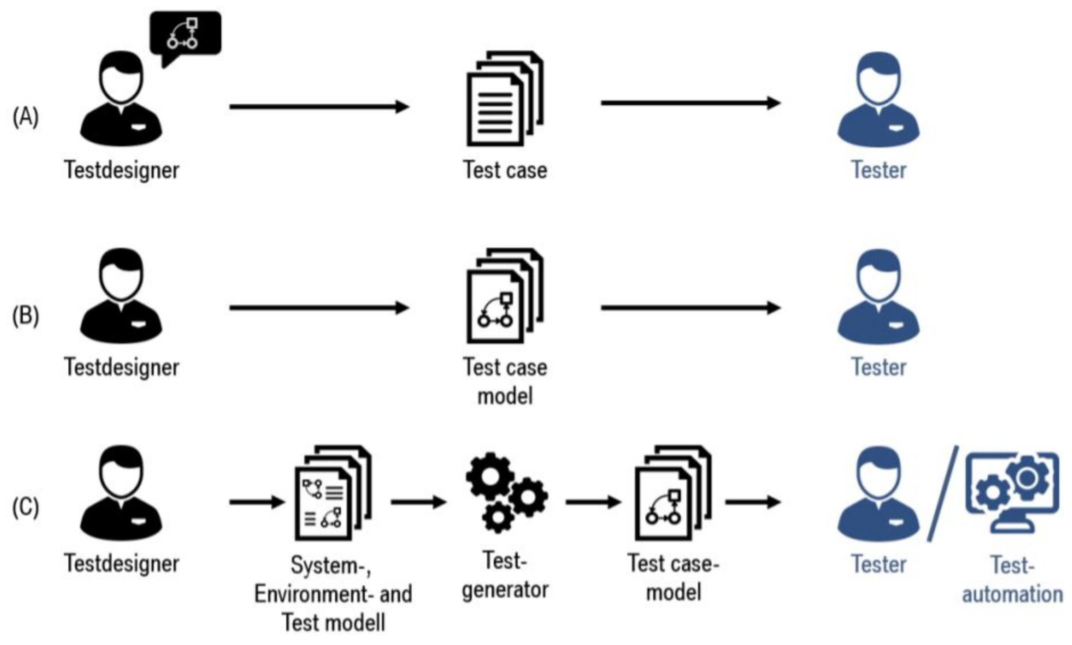


Figure 8.1. Textual vs. model-based specification of tests

8.2 Software-in-the-Loop

The Software in the Loop testing is a test methodology where executable code such as algorithms (or even an entire controller strategy), usually written for a particular mechatronic system, is tested within a modelling environment that can help prove or test the software [19].

The SIL generated code runs on the PC where the simulation takes place. This phase requires only the simulation model and it is hardware independent, the focus being on software interfaces and numerical results. The software requirements and specifications can be analyzed and checked here, requirements refinement starts in this phase during SIL simulations.

A few items typically used in model-based design for BEVs and HEVs are given below. Simscape is an extension of Simulink designed to model multi-domain physical systems. It is used by system engineers to define and build models representing the physical structure of the system.

8.3 Processor-in-the-Loop

The Processor in the Loop is a test technique that allows designers to evaluate a controller, running in a dedicated processor, of a plant which runs in an offline simulation platform [20].

Compared to a SIL simulation, PIL requires the generated code to be executed on the target platform (MCU), making possible a first estimate of the performance on target. Run-time metrics that were not obvious in a SIL simulation, due to the higher computing power of a PC, have a first chance to be noticed here. PIL results should detect platform-related bottlenecks and insufficient hardware capabilities; otherwise it validates the performance on the MCU. Of special interest are the benchmarks, memory content view, disassembly options, hardware interrupt monitoring, waveform analysis, thermal effects and Electromagnetic Interferences (EMIs), among others.

8.4 Hardware-in-the-Loop

The Hardware in the Loop simulation is a technique for validating your control algorithm, running on an intended target controller, by creating a virtual real-time environment that represents your physical system to control. HIL helps to test the behavior of your control algorithms without physical prototypes [21].

Due to its complexity, developing ECU software on prototype vehicles takes time and cost. Considering the implications of a severe powertrain fault, safety played a key role when engineers looked for other design methodologies. Having decades of experience with the internal combustion engines, a strong level of knowledge already existed while switching to other types of motors like HEV or BEV.

Developers rely nowadays on model-based design methodologies, focusing on the operation of a specific component and the overall functionality, rather than writing code from scratch or worshiping a specific programming language. OEMs and suppliers from the automotive industry agree that HIL systems have become the de-facto standard in software engineering, electronics, and mechatronics development. This approach has become mandatory when building large failsafe systems, related to transportation, aerospace, defense, etc.

ECUs are often available sooner than the rest of the vehicle components, enabling a large amount of the functionality to be known before building the car. It is a common practice for suppliers to use HIL systems for validating their individual subsystems and to provide their simulation models in a standard format.

HIL systems can be used for developing and testing a simple software function, the full set of hardware and software ECU functionality and large systems containing mixed types of control units (motor, automatic transmission, stability, etc.). The ECU is supplied with electrical signals, which are the result of a real-time dynamic simulation model. The output signals that it sends in response are read as input variables by the real-time model. In the frame of this approach, a computer running Simulink executes the real-time system models. Therefore, the ECU can be operated in a laboratory environment exactly as it were in a real vehicle. Furthermore, such a test bench allows real-time access to all signals that are difficult to measure in a real system. Operating points and corner cases that are almost impossible – or too risky – to recreate in a real system, can be handled in an elegant way by using HIL simulators. A simple example can be the testing of the

Electronic Stability Program (ESP) while driving at high speed. Single and multiple faults can be injected during simulation, covering the most improbable real-life situations. The costs for fixing errors are much lower if they are found in the early design phases [22].

8.4.1 HIL simulator for electric vehicles

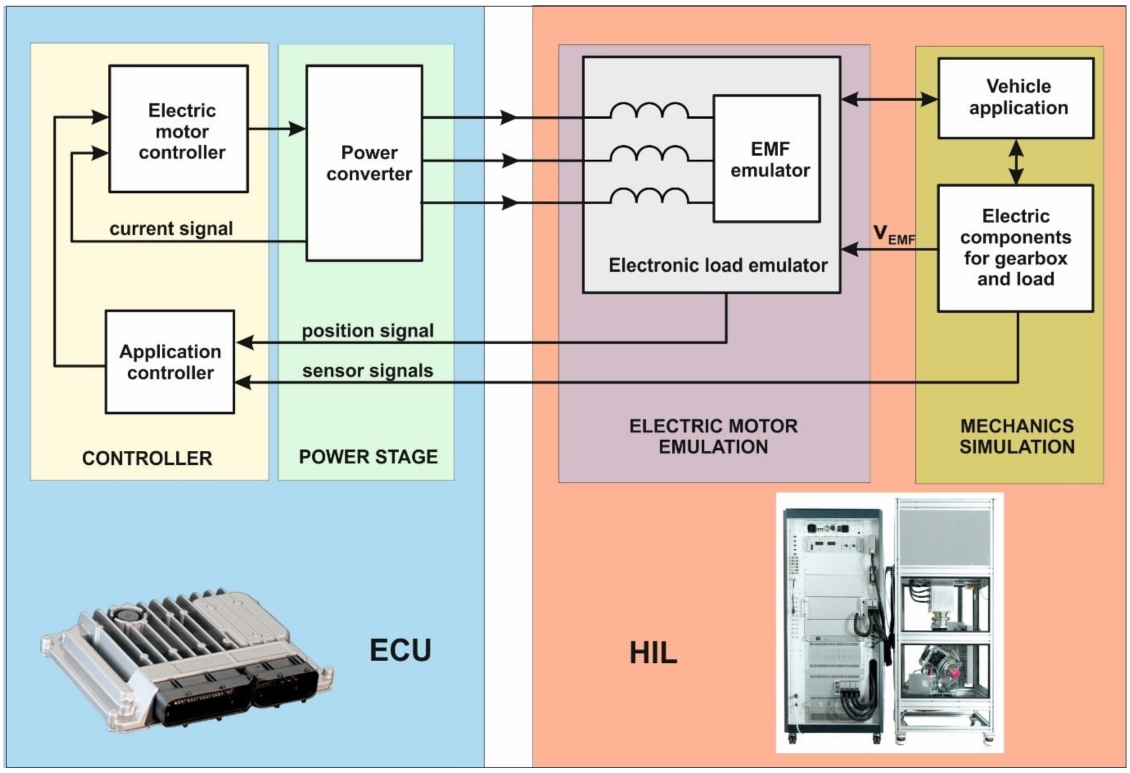


Figure 8.2. HIL diagram for an electric vehicle ECU

Electric motors have a more dynamic behavior than internal combustion engines, requiring ECU clock signals around 10-25 KHz, which is significantly higher than traditional ECU HIL applications for powertrain or chassis domains. Unlike power systems, power electronics converters rely on high frequency controlled and self-commutating switching devices. Power MOSFETs, IGBTs, SCRs and thyristors operate at frequencies in the order of tens of kHz, the required spectrum to analyze their dynamic system effects exceeding 1 MHz, mostly due to interharmonic and harmonic superposition. It becomes clear that a Nyquist frequency of 50 KHz will not suffice for an accurate data acquisition. Given the large number of signals to be processed in real-time by the HIL, a compromise needs to be done. Many systems use a factor of 3. . . 5 for multiplying the Nyquist value, for making sure that most system dynamics frequency content is included in the processing. Therefore, electric motor control applications are simulated in a convenient way with FPGAs. Typically, in a HIL simulator diagram the FPGA supports the motor model and the three-phase inverter model. It covers functionalities like PWM signal measurement,

position/angular sensor simulation, dynamic ohmic losses in switches and diodes, as well as high rate resolvers and encoders.
Besides the control of the shaft speed of the electric motor, it must also handle torque related currents in a fast control loop. This is why the HIL simulator must meet requirements for high power output and current sink.

On the other hand, the mechanical models and parameterization for the FPGA models are handled by a processor board. The simulation models for purely mechanical elements, such as gearboxes, reduction drives and shafts with load can be performed by programmable boards. The sensor signals read by the ECU in the HIL setup are a close match to the real ones, delivered by transducers attached to the mechanical components in a vehicle.

The controller module reads input signals like current from the power converter, position from the electric load emulator, and sensor signals from the electronic boards simulating the mechanical components.

The power converter generates the three-phase signals to be delivered to the electric motor, using the real-time information from the electric motor controller. Two solutions are possible:

- Operating a real drive motor on a test bench.

- Emulating the electric motor.

The second option means reproducing the electrical behavior of a real motor with hardware that does not necessarily looks like one, but contains components with similar functionality, even some configurable software. Compared with the first solution, a purely electric test bench is easier and safer to operate.

Tests can be run at very early development stages, even when the actual electric motor is not available yet. Its physical properties, such as inductivity, torque generation and power consumption are represented with a high degree of realism. It must also emulate accurately the EMF.

The power signal interface – for both ECU and HIL – needs to be defined to cover various use-case scenarios, such as gate drive signal loss or open-phase fault. They point to situations where power stage electronics is faulty or motor windings are physically interrupted. The HIL must provide flexible electronic control interfaces for covering all relevant use-case combinations. It must also allow the reproduction of open-circuits, for example via banana shorting plugs on its panels.

The mechanical interface holds signals related to elements which are not electrical in a real vehicle. Driving speed, vehicle mass, geometry and dynamics, tire radius, gear ratios, steering wheel angle, mechanical load, throttle, braking system, aerodynamics and drag resistance, road grade, can all be simulated here. Being a critical mechanical component, the gearbox needs an accurate simulation model for covering all the relevant transient, frequency and complex frequency phenomena. During iterative design phases, combining its model with the motor's one may lead to optimum parameters for the entire powertrain section [23].

8.5 Safety

ISO 26262 defines functional safety to "absence of unreasonable risk due to hazards caused by malfunctioning behavior of E/E systems" [24][25].

Safety is of great concern in the automotive industry and so is the need to display proof that substantial safety measures has been taken for a certain product. Functional safety and benchmarking are essential concepts for the development of safer vehicles.

8.5.1 Functional Safety

Functional safety is the part of the overall safety of a system or piece of equipment that depends on automatic protection operating correctly in response to its inputs or failure in a predictable manner (*fail-safe*). The automatic protection system should be designed to properly handle likely human errors, hardware failures and operational/environmental stress [26].

According to ISO 26262, functional safety is defined as the "absence of unreasonable risk due to hazards caused by malfunctioning behavior of electrical/electronic systems" [27].

ISO 26262

ISO[1] 26262 is a recently introduced functional safety standard for E/E systems in road vehicles. It embody the full life-cycle of the system, including management, development, production, operation, service, and decommission of the system. Product development includes requirements for initiation, specification, architectural design, unit design and implementation, unit testing, integration, and verification. It uses an ASIL for specifying qualitative and quantitative requirements for safety related functions to be implemented by E/E systems and provides ASIL-dependent requirements for the whole lifecycle of E/E system (including hardware and software product development), as well as for confirmation methods and measures to ensure sufficient safety.

- The system will implement safety mechanism of detecting and monitoring the power supply of the MCU to detect over voltage and under voltage failure modes.
- The system will implement window watchdog safety mechanism to the MCU to detect either clocking or program sequence failures of the MCU.
- A software test to detect latent faults in safety mechanism implemented by HW in the MCU [25].

[1]The International Organization for Standardization **ISO** is an international standard-setting body composed of representatives from various national standards organizations. It creates documents that provide requirements, specifications, guidelines or characteristics that can be used consistently to ensure that materials, products, processes and services are fit for their purpose.

Chapter 9

Test Strategy Line

9.1 AOI

AOI is an automated inspection of the Printed Circuit Board Assembly (PCBA) manufacturing in which a camera autonomously scans the DUT checking for both catastrophic failures (e.g. missing components) and quality defects (e.g. lifted components). It is commonly used in the manufacturing process because it is a non-contact test method.

9.1.1 AOI Equipment

AOI is a test process which allows the operator to quickly and accurately ensure that the PCBA under test have been correctly built without assembly errors. The AOI is a complex system composed by the following elements:

- **Monitor**: It is the mean through which the operator communicates with the machine. From the monitor the operator starts the board surface image acquisition and later checks that it does not present any defect.

- **Lights**: A complex system of lights is used to illuminate different angles of the board with multiple colors such that, various details of the board surface are highlighted.

- **Camera**: The camera is in charge of acquiring the image of the target area and sending it to the processing unit which is ready to be processed by the operator beside the monitor. The better the cooperation of the lights with the camera is more detailed the view of the product surface becomes.

- **Storage/backup**: A storage is needed if a "statistical pattern matching" is performed. This technique exploit a database of older images for comparing the current image with all occurrences of historical camera acquisition.

- **Processing Unit**: It is the core of the system which receives the camera acquisitions and processes the digitized image of the entire board surface, possibly highlighting potential defects. This resulting image is shown up on the monitor allowing the operator to check whether the highlighted defects are actually present on the PCBA or not [28].

9.1.2 AOI Techniques

The board under test is captured several times with different angle positions and using different lights (fluorescent, LED and infra-red/ultra-violet). Then, an image processing technique merge all the photos in order to build up a digital picture of the board. This is a time-consuming operation and there's a speed/accuracy trade off for the image capture system.
After the acquisition phase, the captured image is processed by a complex algorithm that is able to check if the board surface is as expected. This algorithm is based on image comparison, and it is able to discover defects and highlight unwanted situations. The strategies used for comparison of the DUT with a perfect digitized board (*golden board*) are:

- **Template matching**: AOI system uses the "golden board" stored image to make comparisons.
- **Pattern matching**: AOI system compares the actual DUT with a statistic set of stored pattern figures related to both good and bad Printed Circuit Board (PCB) assemblies.
- **Statistical pattern matching**: AOI system compares the DUT with an averaged figure corresponding to a set of previous acquisitions of the same board. The advantage in this case is that the system learns dynamically how to create an averaged image even more precise and reliable [28].

9.1.3 AOI Limits

This technique can be very useful in order to discover physical defects of the board and to avoid catastrophic situations for which a repairing of the board is needed before proceding. It is important to notice that the AOI only assists the operator in recognizing defects since the attention of the user decreases, while a complex system like this is very fast and precise.

9.2 ICT

In-Circuit Test (ICT) is an example of white box testing in which a PCBA is tested by electrical probes checking for shorts, opens, resistances and all other quantities that show whether the PCB was correctly fabricated and assembled.
It can be performed using a test equipment with a bed of nails test fixture, or with a fixtureless in-circuit test setup.
This technique is able to discover defects like:

- PCB integrity and functionality.
- Passive components soldering and positioning.

- Passive components value and tolerance check.
- Active components mounting and basic functionality, when possible.

This technique is not intended for testing board functionality, but it checks whether each component is in place and if it has the correct value [29].

9.2.1 ICT Equipment

A common ICT system is composed by the following elements:

- **PC**: The PC is used to handle the operation concerning the ICT test to be performed (run, check results, program the test sequences to be applied and so on).
- **ATE**: The Automated Test Equipment involves all equipment needed to perform the test of a board such as the analog/digital instruments, power supply blocks and channels to connect the ATE instruments to the ICT tester nails.
- **Bed Of Nails**: It is an electronic test fixture with an high number of nails that directly touch test points of the DUT. Each nail is multiplexed to an ATE instrument in order to stimulate and sense a signal to/from the DUT. All test points of the board can be contacted at once (it exists a Flying-Probe variant with few nails moving dynamically on the PCB surface). In an idle position nails do not contact the board. However, when pressed down, the DUT is accessed through access points/nails contact.
- **ATE/Bed of Nails communication system**: Nails should be multiplexed with precise instruments within the ATE. A set of programmable system of relays and switches is programmed ad-hoc for each DUT. Once that test instruments are multiplexed to the channels, a final step called "scanner" is programmed by picking up signals coming from channels and bringing them to the desired nails in order to stimulate or sense precise elements of the board [29].

9.2.2 ICT Techniques

The whole ICT flow is composed by several sub-tests performed in order:

- Pre-Screening Test: This test search short, open and non-contacting nail defects on the entire board.
- Passive Component Test: This test is performed with powered-off board in order to detect the presence and to measure the value of some components such as resistors, capacitors, inductors, diodes and IC components presence.
- Discrete Component Test: This test is performed with the powered-off board in order to detect the discrete behaviors of certain components (e.g. transistors).

- Special Test: Special tests are dedicated for those components that require particular stimulus and measurements scenario to be performed, different than commonly used routines.

- IC Test: The supplied test requires powered-on board in order to verify the functionality of active components.

The ICT standard flow uses the "growing power" technique which provides that first phases are characterized by low level of voltages which step by step grows up until reaching last phases (where usually Vbatt is reached).
An exception is made for the special tests since the level of voltage depends on unit to be tested and adopted techniques.

The ICT flow is a sequence of routine calls. There is an executable program, running on ATE, that launches these routines to stimulate and sense board nodes. If all sub-tests pass, the board is marked as "good" otherwise, at the first fail, the board is marked as "faulty" and it is discarded.

Looking at the ICT flow, it can be noticed that there are often two particular routines:

- **Discharge Capacitors**: This operation is needed to avoid problems due to residual charge that provide injection of currents (it is needed a certain amount of time in order to completely discharge capacitors).

- **Stop On Error**: This operation is a safety-choice due to the fact that moving forward after a failed sub-test can compromise the integrity of the board [29].

9.2.3 ICT Detection Capabilities

An important aspect in the ICT technique is that the unique means through which ICT nails can contact the DUT are the Test Points specifically designed to the board (this limits the tests that can be performed).

9.3 EOL

Functional Test is performed as a final step during the so called End of Line. It provides a PASS/FAIL determination on finished products, before they are shipped to the customers. The final product is subject to a validation process in order to ensure that the HW is free from defects that could, otherwise, adversely affect its correct functioning in a system application.
Before flushing the application software to the board, the EOL validates the product functionality to attest that it is definitely ready to work in field.
It is worth noticing that the requirements of the functional test, its development, and procedures vary widely from product to product and from system to system.

9.3.1 EOL Equipment

The logic heart of an electronic board is an MCU. The whole board, communicating with the external environment through the connector, receives some input signals which are processed (by some input stages) and presented to the MCU which, in turn, decides what are the actions to take. Signals coming out from the MCU are processed by some output stages in order to be presented to the connector (and so to the external environment). This is the operational mode of the board.

During EOL the MCU is not flashed with the application software yet. It is so a virgin IC without a "brain". The **Test Firmware** has the role to provide to the MCU a certain intelligence in order for the EOL test to be assisted and supported. In this way it is possible to stimulate and read MCU signals from the connector with a precise protocol. During the test, the hardware equipment (ATE) is in charge of emulating the real environment in order to send and receive signal to/from the DUT with the purpose of ensuring that the DUT is intact in all its functionalities.

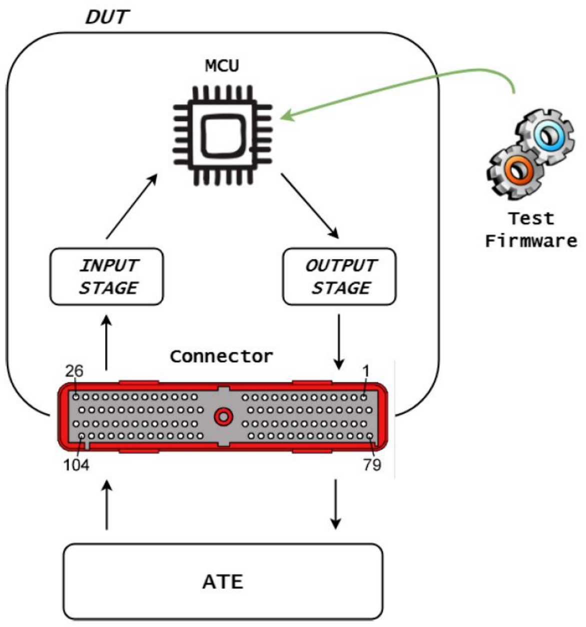


Figure 9.1. EOL - Test Firmware

9.3.2 Test Firmware

The virgin ECU is flashed with a firmware and a powerful ECU test software called On Board Test (OBT) which is loaded into the board. The OBT software is used for different test purposes in three different ways:

- **EOL mode**: This is a client/server test mode used in coupled plants with an EOL ATE.

- **Power Run In mode**: This is a stress profile test mode for plant purpose coupled with Power Run In ATE.
- **Burn In Self Test mode**: This is a stress profile test mode for quality purpose used for process validation tests.

The purpose of EOL test is to check the HW integrity of the ECU and to verify the FW basic functionalities.

The ATE can set the OBT in EOL mode, then the OBT starts by executing commands sent from ATE (sensors acquisitions, actuators commands and diagnosis, Communication line test, Safety test).
The most difficult part of the EOL testing is to find the highest number of functions of the DUT that can be tested and how to test them. This is an aspect which varies from company to company and depends on the Testing Team's know-how and past experiences.

EOL testing needs the utilization of big, expensive, powerful ATEs. The ATE hardware constitutes an emulation of the hypothetical real environment in which the DUT should work. This environment is the union of instruments and Loads. Of course all these aspects are strictly correlated, and all have to be improved in parallel, otherwise the worst one creates a bottleneck in the capability exploited by the test.

9.3.3 EOL Techniques

The EOL testing flow doesn't have a prefixed structure. The role of the Test Engineer is developing functional tests based on a series of technical supports:

- **Methodology Book**: This is a kind of "Lesson to Learn" for the whole team. It is a continuously updated reference point which describes in a generic way (independent of the adopted test equipment) how to perform certain tests. The notions inside this documents are very technical. The role of the Methodology Book is to instruct the Test Engineer and give him an idea of what has to be done to test certain situation.
- **Prescription**: It is a specification which involves the technical details of all instruments needed for performing the test (what you need from ATE, what you need in terms of load to simulate the external environment, what you need in terms of firmware and so on). It is created by the Test Engineer that, after a study of the electrical schematic and the the Methodology Book, defines all specific details that are involved in that tests.
- **File Sequence**: It is the sequence of all the tests applied to the board during EOL. The software used to describe the sequence (TestStand) creates a layer of routine calls. In turn, the routines calls device drivers of the ATE hardware instruments.
- **Datalog**: It is the output document produced by the ATE at the end of the functional test, it reports technical details about the test steps and whether

they have PASSED or FAILED. This document is very important during the mass production phase since it is interpreted by the operator in order to decree whether the product has to be discarded (in case of FAIL) or sent to following phases in the production flow (in case of PASS).

The common flow followed by the Test Engineer is the following one:

1. **ECU Study**: The very first step is to carefully analyze the Electrical Schematic to understand what are the main functions implemented by the board how to test them.

2. **Prescription Drafting**: After study phase the Test Engineer is aware of most of the technical details involved (such as voltage and current involved, ranges to be compared with read values, characteristics of the instruments used and so on) and the prescription is drafted.

3. **Test Sequence Drafting**: After having defined the technical details of the steps to be performed, the real test sequence is created. The DUT is tested in different operating modes.

4. **Test Sequence Debugging**: Together with the Test Sequence Drafting, a debug phase is performed. It consists in tuning the test sequence to make it more and more stable till reaching the final version.

9.3.4 EOL Limits

The EOL limits are mostly related to the precision in the measurements of the component value. The following description is referred to a resistor, but it can be extended to any kind of component.

The measurement principle to test a resistor is the same as the one used by ICT. However, ICT can directly access to the resistor pins (as far as the test points are present), whereas EOL follows a path that starts and ends from/to the connector pins traversing the input/output stages and the MCU which operates according to the firmware directives. In addition to this path (which impedance is known and computable from the schematic), an additional critical impedance (whose value is uncertain due to deterioration and impossibility to be reconstructed) is introduced by the ATE in the complex interconnection wiring system between instruments and panel.

Part III

CASE OF STUDY

Chapter 10

The work with Marelli

The first and second part of this thesis are mandatory in order to know the key elements on which the Magneti Marelli-TTE team is working on.

My initial role inside the company was to carefully analyze the general structure of the team and to meet different team members in order to see and learn what they do, and how they do that. Another important point is learning the Software and the proprietary Hardware used inside the company. Not only purchased equipment, but also a lot of intellectually proprietary HW are present inside the company. Many hours of work were spent by Magneti Marelli engineers in order to design and realize powerful HW modules that are used for ECUs testing purpose.

It is also important to note that the test operations begin in parallel with the developing of the first prototype of the ECU. The target ECU is subject to several modifications and improvement before having a final perfectly working ECU, ready for the mass production.
Each time a new prototype is released it is necessary to carefully check what have been changed and what needs to be tested again. Also some modifications has to be done in the testing FW for the new ECU version since different prototypes can differ in HW and in the design.

10.1 IBM Rational Synergy

Magneti Marelli is a large company with a huge number of engineers working on the same projects, so all them need to be synchronized and stored in a big database.

There's a task-based Software Configuration Management (SCM) that brings together global distributed development teams on a unified platform. The single SCM repository manages all artifacts related to SW development, including source code, documents, the final software executables and libraries.

The software used is IBM Rational Synergy, and it can be accessed only by authorized users. Each user has a well-defined role and, according to the role, an user can create a new project, or a personal version of an existent one in order to split the development, and work in a dedicated version of the project in parallel with other users.

About my work, I had a developer account to create projects and to branch existing ones. My training with this software provided the creation of a new task, based on an already closed project. Closed projects are totally working, and they are available for new developments made by other people in the team. Each team of the TTE has a well-defined role. They add part of code in the project in order to perform the required task before closing it and make it available for the next team.

10.2 Compiling a project

Each project comes with some make-files written ad-hoc by engineers from the company. The compiling operation takes time and it is done by connecting to a remote powerful server in order to speed up the entire process. A single mistake in one of the thousand files composing the project can be the cause of several errors during compilation time.

I've worked with a 3 core MCU (Infineon), so it was necessary to compile each core once before compiling the whole project. Once the compilation phase is done the program can be downloaded into the target board that has to be connected to a debugger tool.

The company uses a Lauterbach debugger tool directly connected to the target ECU, which is accessible online through a static IP address.

The basic sample project consists in an MCU running with a dummy task that increments a counter at each execution of the task. The debugger is able to sense the variable changing and it shows that it's incrementing according to the specification. The OS supports the execution of three tasks at different frequency, slow speed, middle speed and the last one at the maximum speed allowed by the MCU.

This is the starting point for the developing phase. Once ensured the FW is working and it fully compiles, it's time to start writing test programs.
The FW designer, during its work, writes some ad hoc functions and modules capable to mange all the HW exposed by the target board. These modules can be used by the engineers for testing purpose since they are able to manage input/output analog/digital voltages/currents and all HW interfaces available on the target.

10.3 Device Drivers

The FW which runs on the target board is not an application FW, but it's a testing FW. This means that the test engineer uses a special FW that permits the access to the whole MCU by ignoring several constraints that are present at the application level. This is important because during the testing phase there will be tested things (operations and configurations) that shouldn't take place during the application phase, but they have to be tested as well. The firmware test contains a set of smart interfaces that could be used in order to manage all the HW present inside the target board. These interfaces are called Device Drivers (DDs) and they are written ad-hoc in order to stimulate and test each single pin and component present on the target board. It is possible to use some predefined functions for initialization, configuration, write values, read values and so on, depending on the module chosen.

10.3.1 DD OutPwm

Let's analyze a common DD for managing a Pulse Width Modulation module. The test FW exposes some functions that must be called in order to communicate with this peripheral:

- DD_OutPwm_Config(Idn, ROM *CfgDyn)
- DD_OutPwm_Get(Idn, RAM *Data)
- DD_OutPwm_GetDiag(Idn, RAM *DiagData)
- DD_OutPwm_Put(Idn, Duty, Cmd)

In order to use this PWM module it's necessary to fill a CfgDyn structure with the desired configuration constraints. Then by using the function *DD_OutPwm_Config()* the module is enabled and configured.
After the configuration phase, the module can be enabled by defining the duty cycle and calling the function *DD_OutPwm_Put()*.
It is possible to see that the PWM module is correctly initialized and activated through some drivers proprietary of Magneti Marelli. There are some specific configurations to be done in order to make it correctly working. Having an universal way for enabling a PWM module independently of the target ECU could be a good point.

10.4 UDS integration

All the work done up to now inside the company is related to proprietary HWs and SWs. The interfaces were all internally made and fully customized by test engineers in order to fit the ECU characteristics.
The next step was to use the UDS standard, that uses CAN bus, in order to internally call some functions that today are called with proprietary functions.
Since UDS is a standardized communication protocol, the idea was to create a more flexible architecture that could fit more or less all the ECUs. Each of them have a CAN bus protocol. The idea was to use this protocol to adapt higher level calls (UDS) by redirecting them to lower level drivers (HW). In this way the high level remains unchanged for all the ECUs, and the low level is the only one that needs some small adaptations to perfectly fit the target ECU.

10.4.1 ECATE

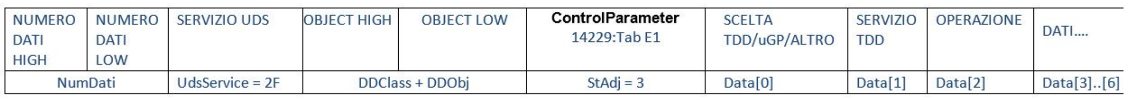
ECUs have a communication layer to manage the communications. A complete communication layer can also manage UDS messages coming from CAN bus.

Since UDS is a standard protocol, it is sufficient to follow the ISO specifications for creating a stream of data to be transmitted. The communication layer filters UDS messages that are non compliant with the protocol, and it unpack messages accepted.

One of the projects of company is the development of a new FW version able to exploit the standard UDS protocol for calling proprietary DD modules for testing purposes.
This project is named **"ECATE"** and it is based on a standard project with some modification in the communication layer in order to accept also the 'hidden' calls to DDs modules done through UDS messages.
The idea is to filter the UDS message as provided by the protocol; in case the message is related to a DD call, it is unpacked and redirected through the right path.

In order to better understand what is the basic idea, let's analyze the UDS service *$2F* also called 'Input Output Control By Identifier'.
"This service allows an external system intervention on internal/external signals via the diagnostic interface. By specifying a so-called option bytes, additional conditions for a request can be specified" [30].

My work with the development of ECATE is based on the realization of the routing of an UDS message for accessing a DD. An UDS transmission is made by a single stream of data. These data will pass through a filter. If the service requested is known, the data is accepted and then unpacked to recover the message sent. With a single long stream of data it is possible to call multiple functions of DDs. It's sufficient to know how to split the data.



NUMERO DATI HIGH	NUMERO DATI LOW	SERVIZIO UDS	OBJECT HIGH	OBJECT LOW	**ControlParameter** 14229:Tab E1	SCELTA TDD/uGP/ALTRO	SERVIZIO TDD	OPERAZIONE	DATI....
NumDati		UdsService = 2F	DDClass + DDObj		StAdj = 3	Data[0]	Data[1]	Data[2]	Data[3]..[6]

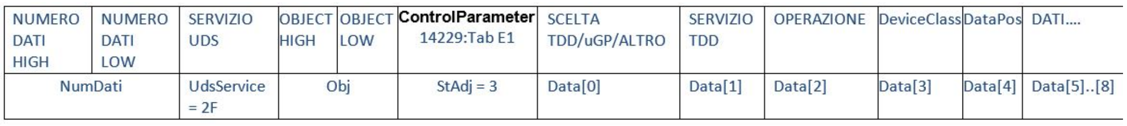
Figure 10.1. UDS message, frame division

The Figure 10.1 shows the frame division of the UDS message. There's an ECATE communication protocol that can unpack DATA in order to know which DD to call. This protocol consists in a series of if-else conditions checking for different fields:

- DATA[0] -> ECATE Service
- DATA[1] -> DD Service
- DATA[2] -> DD Operation
- DATA[3] -> Device Class

- DATA[4] -> Data Position

During the development, a problem came up concerning the communication layer. There's a limitation of 10 bytes as maximum size of UDS messages. It is not possible to send messages longer than 10 bytes. This problem is a 'work in progress' and will be fixed with the next FW release.
A temporary modification is done in the UDS message to overcome the COMM layer problem. To continue the development I send multiples short UDS messages instead of a single long data. The Figure 10.2 shows the modified UDS frame division. An additional field *'DataPos'* is needed to know which data we are receiving before the recomposition of the DD call.



NUMERO DATI HIGH	NUMERO DATI LOW	SERVIZIO UDS	OBJECT HIGH	OBJECT LOW	**ControlParameter** 14229:Tab E1	SCELTA TDD/uGP/ALTRO	SERVIZIO TDD	OPERAZIONE	DeviceClass	DataPos	DATI....
NumDati		UdsService = 2F	Obj		StAdj = 3	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]..[8]

Figure 10.2. UDS message, modified frame division

Following the example of PWM described before, we can analyze the corresponding UDS function aiming to enable a PWM module.

Code needed to fill the structure for dynamic_configuration():

- 0000 0300 4003 2300 0000 0001 (Enable)
- 0000 0300 4003 2301 0000 0000 (R_Excep)
- 0000 0300 4003 2302 0000 0000 (Excep)
- 0000 0300 4003 2303 0000 03E8 (Period 1000us)
- 0000 0300 4003 2304 0000 0000 (Mode: PWM)
- 0000 0300 4003 2305 7000 4D8A (Variable no-fault)
- 0000 0300 5000 2300 0000 0000 (Call Config())

Code needed to fill the structure for put():

- 0000 0300 4004 2300 0000 01F4 (Duty cycle 50%)
- 0000 0300 4004 2301 0000 0000 (Command: not needed for PWM)
- 0000 0300 5003 2300 0000 0000 (Call Put())

The UDS service $2F gives back a response message. We exploit the response message in order to check that the requested operation is successfully done and, of course, we can check that the HW pin, stimulated by our call, changes its value as specified [1].

[1] *"ECATE"* material supplied within the company.

MMARTE

The UDS message is a buffer of data sent through CAN bus to the target device. There are several ways for sending a CAN message through a PC.
Inside the company, testing team uses a proprietary SW called **“MMARTE”** in order to packet the data to be sent through CAN bus (at SW level).
The data to be sent pass through a VECTOR CAN TOOL that is in charge of sending the data respecting the CAN bus protocol constraints at a low electrical level.

10.5 Infineon AURIX™ TC3xx

This prototyping board powered by a TriCore TC3xx microcontroller will be the heart of several future projects inside Magneti Marelli since the old ECUs were using the past Infineon family TC2xx. With this new generation there will be several advantages in particular for the safety field. Safety is very high level but using of AUTOSAR is mandatory in order to maintain the security level reached by the vendor of the board.
Using AUTOSAR also means using a completely new architecture, very different with respect to the one used till now inside the company. Using or not AUTOSAR is not a choice because a tier1 supplier like Magneti Marelli is subject to customers requests, and it has to respect the customer requirements for the ECU.

“With its up to hexa-core high performance architecture and, its advanced features for connectivity, security and functional safety, the AURIX™ microcontroller TC3xx family is ideally suited for a wide field of automotive and industrial applications. In addition to engine management and transmission control, targeted powertrain applications include new systems in electrical and hybrid drives. Specifically hybrid domain control, inverter control, battery management, and DC-DC converters will benefit from the new architecture.

The AURIX™ TC3xx microcontrollers are well-suited to safety-critical applications ranging from airbag, braking and power steering to sensor based systems using radar or camera technologies. The combination of performance and a powerful safety architecture makes the family ideal fit for domain control and data fusion applications supporting the next levels of autonomous driving.

The latest AURIX™ TC3xx microcontrollers are also well-suited for safety-critical applications to support clean, autonomous and connected cars. Ranging from classic airbag, braking and power steering to fail-operational systems supported by sensor-based systems using radar, LIDAR or camera technologies.” [31]

Safety is a key point of this product, the TC3xx has 2 lock-stepped cores and 2 non lock-stepped cores[2], providing up to 1350 DMIPS in ASIL-D and 1350 DMIPS

[2]**Lockstep** systems are fault-tolerant computer systems that run the same set of operations at the same time in parallel.

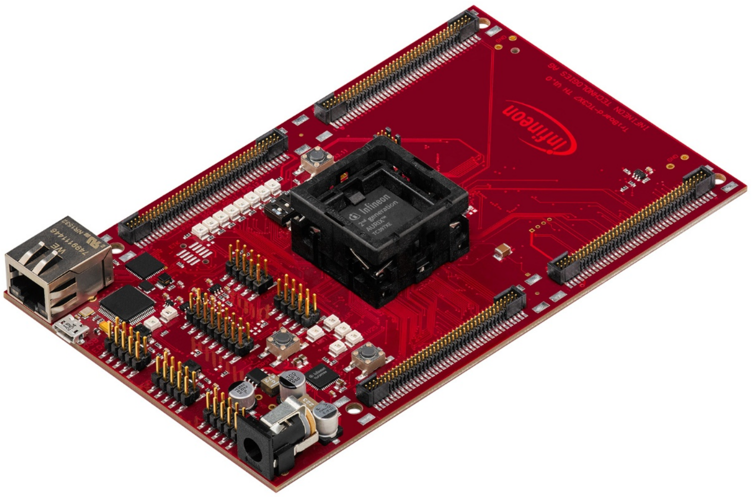


Figure 10.3. TC397 - Developer Board

in ASIL-B [3]. Note that for safety critical automotive applications, it' very important to ensure a safety platform supporting ASIL-D ISO 26262 (the ASIL level constraint depends on the application).

10.6 VECTOR DaVinci

The goal of my work with AUTOSAR architecture was to successfully compile a full systems in order to be loaded inside a target ECU. The system contained also some SWCs that can directly access MCAL interfaces and can drive the microcontroller modules for testing purposes.
The first difficulty in this operation was having a fully compilable project since the integration of external SWCs with the basic architecture of an ECU was a delicate operation and it needed several configurations.
Configurations are related to the creation of application components, port prototypes, and relatively realization and scheduling tasks.

For the development in the AUTOSAR environment, numerous meetings were held with the various SW companies. Several purchase offers were evaluated. In the end it was chosen for the Vector DaVinci SWs and several training courses were held for engineers in the company.
At the time of my arrival there were no ongoing training sessions so I relied on paper material provided by the company. In particular I worked with this two tools:

[3]**Automotive Safety Integrity Level (ASIL)** is a risk classification scheme defined by the ISO 26262 - Functional Safety for Road Vehicles standard. The ASIL is established by performing a risk analysis of a potential hazard by looking at the Severity, Exposure and Controllability of the vehicle operating scenario.

- DaVinci Developer: for creation of SWCs description, and for definition of SWCs internal behavior.
- DaVinci Configurator Pro: for creation of ECU configurations and generation of configurable part of BSW and RTE.

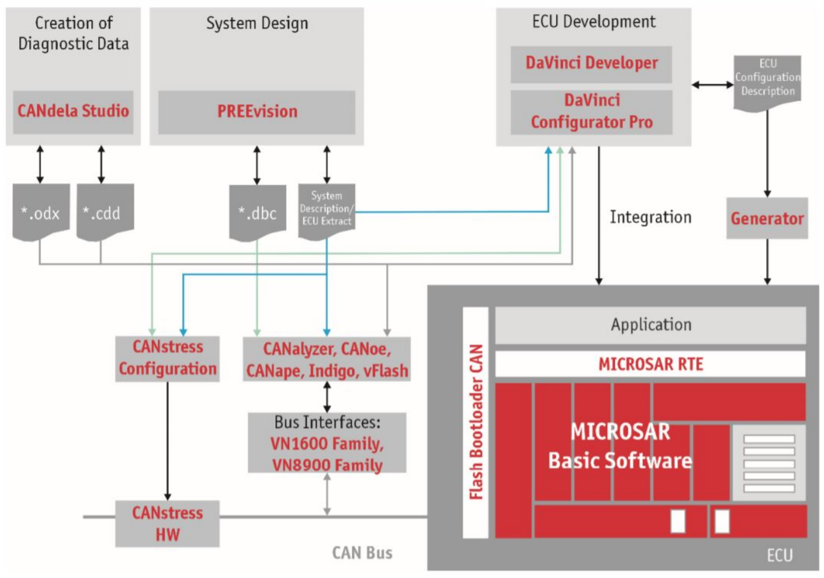


Figure 10.4. Developing AUTOSAR ECUs

Both tools operate on the same project (.dpa), where the DaVinci workspace (.dcf) is shared among the tools. A mutual locking mechanism ensures that the same project file can be safely opened with both tools.

The operating systems is called **MICROSAR** and it provides MCAL basic software ant RTE (Figure 10.6).

The main idea of using AUTOSAR is moving the focus to the configuration phase instead of writing lines of code. The huge work of writing hundred lines of code passes to the Configurator Tool that has to be carefully configured for each specific project.

Usually the basic configuration of each project is buyed by the vendor. The OS is provided with all necessary modules already configured and ready to use for developing purpose.

My case of study was the realization of a simple SWC for switching *off* and *on* an HW pin of the target board. Of course the HW pin has to be right mapped inside the low level MCU configuration.

For the purpose of my tests I choosed an HW output pin accessible out of the target board. Remember that the target board contains the microcontroller embedded with a very complex circuitry around it, so touching (changing value) of some pins may cause some misbehaviours. Some pins could be also internally driven, so driving such pins externally may give an undefined behaviour due to some internal computation of the target board.

Knowing these limitations, after having carefully analyzed the schematic of my target ECU, I choosed these two pins:

- KillSw_ID (DIGIN0104) - digital input pin
- ModSplyEn_OD (DIGOUT1511) - digital output pin

These two pins are easily accessible by the target board, and they don't compromise the internal behaviour of the board.

The first important point is to map the selected pins inside the Vector Configurator Tool. AUTOSAR have an MCAL and a SWC of type *Complex Device Driver* is able to directly access the MCAL.
An important point is that, having an hybrid AR-MARELLI architecture, it's necessary to configure AR in order to point to a pin which is already internally mapped to a Magneti Marelli-Device Driver inside the classic MARELLI project.

Following this idea it becomes possible to interact with all the HW present on the target board by using the existing MM Device Drivers, and this pins can be driven by an high level AR-SWC.
From the testing point of view, the first testing purpose is driving digital pins of the target board. This first step is reached with my application.
This starting program will be the base for the future developing since the configuration phase is done now, and later it will be necessary only to add other SWCs for driving IO ports, or modify the existing SWC specifying another IO port to drive.

Part IV

CONCLUSIONS

Chapter 11

Conclusions

The work done during the internship period within the host company was very formative and interesting from the growth and knowledge of the automotive and testing point of view. Theoretical notions were acquired, not covered during the chosen degree course, also knowledge was acquired about how to work in a company, and how to relate to colleagues and customers.
I worked side by side with testing engineers, FW and SW architects in order to understand how to work in a team and how the tasks are divided. New programs and proprietary tools have been developed within the company and commercial software tools such as Vector DaVinci for developing AUTOSAR ECUs have been widely used.

The aim was to analyze and develop ECU testing techniques in the automotive field. This is a constantly evolving world that must necessarily follow the progress of the automotive vendors in order to be able to satisfy customers increasingly precise requests. The ECU architectures are constantly growing in complexity and quantity of drivers and electronics and the testing phase must therefore keep up to test these ECUs. The various architectures used by the company in the past have been studied, and the changes and improvements that have been made over the years have been carefully analyzed to have a more solid and functional architecture. The testing field has been analyzed where the test engineer has access to a whole series of IO pins that the end user will never see, but which are fundamental in the test phase to guarantee a fully functional ECU in line with customer requests. All the test phases have been analyzed, which take an ECU up to the EOL where it is functionally tested through function calls performed with a ad-hoc written test software.

In the final part of the thesis is analyzed the future environment for the realization of the ECU, and consequently the merits, defects and motivations of the transition from the Marelli world to the AUTOSAR world. Obviously, an AUTOSAR ECU also requires an AUTOSAR-compliant test environment and then a software study and development was carried out in this environment to be able to manage the lowest HW level through a simple SW application.

The goal for effective transition to the new AUTOSAR architecture with regards

to testing involves the introduction of the UDS communication protocol that works via the CAN interface. The messages to be transmitted for the purpose of testing are specific messages, not usable at the application level, which will be traced to the call of specific Device Drivers and HW pins to allow the testing of most of the drivers present on the target.
Currently, the Marelli architecture allows to do all this using the CAN communication protocol and the ad-hoc UDS protocol is being integrated for test calls (ECATE project).
In parallel, the same system is being developed for the AUTOSAR standard. During the months of internships spent within the company, these themes were worked on but a complete, stable, ready-to-produce architecture was not yet built. This is a work that will continue over the months up to a refinement of the algorithms and methods used in the testing team for the realization of test software for ECUs.

The choice to use a standard UDS protocol is a good starting point, as it allows to use a standard that is already present at the communication level, customizing it with the receipt of specific calls available to the testing team. Similarly, the choice to switch to an AUTOSAR type architecture is a sensible option, even if totally different from the standard that has always been used in Marelli.

Nowadays, AUTOSAR is a strongly necessary environment and it is requested by many customers, so a company like Marelli cannot risk staying out of this world. It is therefore important for the company to try to change the way of working considering the new perspective of the AUTOSAR world. The efforts and hours spent to migrate the ECU architecture will be repaid over time, starting to exploit the potential and key features of this standard.

Bibliography

[1] *Magneti Marelli - History*. 2019. URL: www.magnetimarelli.com.

[2] *Magneti Marelli & Calsonic Kansei*. 2019. URL: www.magnetimarelli.com.

[3] *Magneti Marelli - Company*. 2019. URL: www.magnetimarelli.com.

[4] *Powertrain*. 2019. URL: https://en.wikipedia.org.

[5] Magneti Marelli. *TTE Overview*. 2019.

[6] *Anatomy of a battery electric vehicle*. 2019. URL: https://x-engineer.org.

[7] *(H)EV - Main inverter*. 2019. URL: www.infineon.com.

[8] *Technologies and Components for Designing Electric Vehicles*. 2019. URL: www.avnet.com.

[9] *In-Vehicle Networking*. 2019. URL: www.embitel.com.

[10] *Vehicle bus*. 2019. URL: https://en.wikipedia.org.

[11] *AUTOSAR - History*. 2019. URL: https://en.wikipedia.org.

[12] *AUTOSAR - Layered architecture*. 2019. URL: www.autosar.org.

[13] Massimo Violante. "Autosar Basic" *MARELLI Internal Course*. 5.2019.

[14] *Diagnostic Communication Manager Module In AUTOSAR*. 2019. URL: http://sandeeptiwari.com.

[15] *Diagnostic Event Manager In AUTOSAR*. 2019. URL: http://sandeeptiwari.com.

[16] Magneti Marelli. *ECU Device Drivers Level*. 2019.

[17] Paolo Nigro. "Course COMM" *MARELLI Internal Course*. 5.2019.

[18] *UDS Protocol*. 2019. URL: www.piembsystech.com.

[19] *Software-in-the-loop testing applications*. 2019. URL: https://www.add2.co.uk.

[20] *Processor-in-the-loop and hardware-in-the-loop simulation of electric systems based in FPGA*. 2019. URL: https://ieeexplore.ieee.org.

[21] *Hardware-in-the-Loop (HIL) Simulation*. 2019. URL: https://it.mathworks.com.

[22] *Hardware-in-the-loop*. 2019. URL: https://en.wikipedia.org.

[23] *ECU design for electric vehicles*. 2019. URL: https://www.fortech.ro.

[24] *ISO 26262*. 2019. URL: https://en.wikipedia.org.

[25] *ISO 26262*. 2011. URL: www.iso.org.

[26] *Functional safety*. 2019. URL: https://en.wikipedia.org.

[27] Cadence. *Functional Safety Methodologies for Automotive Applications*. 2019.

[28] *Automatic optical inspection, AOI systems*. 2019. URL: www.electronics-notes.com.

[29] *In-circuit test*. 2019. URL: https://en.wikipedia.org.

[30] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *DRAFT INTERNATIONAL STANDARD ISO/DIS 14229-1*. 2011.

[31] AURIX Microcontroller based on TriCore. 2019. URL: www.infineon.com.