Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN

INGEGNERIA ELETTROTECNICA

Ciclo XXIX

Settore Concorsuale di afferenza: 09E2

Settore Scientifico disciplinare: INGIND/32

MULTI-DRIVE POWERTRAIN CONFIGURATIONS FOR ELECTRIC AND HYBRID VEHICLES

Presentata da: **Ing. Davide Pontara**

Coordinatore Dottorato

Chiar.mo Prof. Domenico Casadei

Relatore

Chiar.mo Prof. Claudio Rossi

Esame finale anno 2017

Table of contents

Preface

In recent years, the considerable interest towards the electric vehicles (EV) and hybrid-electric vehicles (HEV) pushed the major manufacturers to invest an increasing amount of resources into the development of these systems, while also allowing new companies to enter the market with innovative solutions.

The reasons behind this have to do, on the one hand, to the ever more stringent environmental policies that have in fact forced manufacturers to seek alternative solutions to improve conventional internal combustion engines (ICEs) or to substitute them altogether.

On the other hand, the economic downturn of the past few years encouraged the spread of vehicles with alternative powertrains because, in addition to having a lower environmental impact, they present lower operating costs compared to conventional vehicles, although in the face of a higher purchase cost.

Moreover, governments often foster, by means of incentives, the purchase of low emission vehicles, among which are obviously counted the electric and hybrid vehicles.

Under the technological point of view, developments in the field of electric and hybrid vehicles led to novel powertrain configurations, thanks to advancements in the battery packs and chemistry [1], [2], conversion components and control algorithms.

This work proposes technical solutions for electric powertrains, which could be introduced into the market with lower investments and production costs with respect to the EV solutions currently developed and sold by primary carmakers. Solutions proposed in this thesis foster a new industrial approach to EV manufacturing, allowing small and medium size carmakers to be competitive on the EV market.

A relevant industrial application of the technologies developed in this thesis is the remanufacturing of existing thermal engine cars by substituting the ICE powertrain with a new full electric powertrain. The “retrofit” was not initially considered at the beginning of the work, but during the years has become of increasing interest and probably, at the end of the work, represents the most promising application of the developed technologies. In these terms, the thesis plays an important role and represents a new approach towards application of circular economy principles to the EV sector.

This thesis will focus on traction systems in which the propulsive force is divided among several drives, or even supplied by different powertrains, whose power stages are independent from each other.

The research activity carried out by the Ph.D. student is included in the *Design of Work* (DOW) of the FP7 European Project n. 604766 denominated “AMBER-ULV”, *Automotive Mechatronic Baselines for Electric Resilient Ultra-Light Vehicle*, topic “GC.SST.2013-3”: *Future light urban electric vehicles*.

The AMBER-ULV project aims to develop a *Battery Electric Vehicle* (BEV), in the compact car segment, by integrating several innovative concepts, resulting from successfully completed R&D projects, and giving a socially acceptable answer to safety concerns without penalizing the driving experience [3].

This manuscript, albeit with the aim of providing an overall description of the topics covered by the research activity, contains many references and links to the objectives of the abovementioned European project. Its goal is to develop a "*dual-drive, dual-battery*" traction system, with high-level control algorithms automatically generated through model-based software design techniques, and executed on a centralized control system based on a real-time (RT) platform, with the capability of integrating multiple algorithms provided by other partners of the consortium.

The dissertation is organized as follows:

In Chapter 1, the advantages of *multi-drive* powertrain configurations are discussed.

The main drawback of *multi-drive* powertrains, that is the control complexity, is mitigated in Chapter 2. It is also described the control algorithm developed for a "*dual-drive, dual-battery*" electric powertrain.

In order to validate the control algorithm and apply it on the real vehicle produced during the *AMBER-ULV* project, the "*Model-Based Development*" was studied by the Ph.D. and deployed as presented in Chapter 3.

Chapter 4 outlines the integration process the Ph.D. student researched within the *AMBER-ULV* project, as needed to make the prototype work for the official validation tests, which are described in the following Chapter 5.

Chapter 1
Multi-drive Powertrains

1.1 Powertrain power sizing

In this chapter, a sizing example for a vehicle powertrain is shown. The case study refers to a compact car, whose typical characteristics are summarized in Table 1.1.

For the given vehicle, the approach to determine the powertrain design power consists in applying the "*Road Load*" equation (Equation 1-1) obtaining a simple model of vehicle power losses, and simulating it with reference to the "*ARTEMIS*" standard cycles.

Once the design rated power is chosen, the feasibility of developing a *multi-drive* powertrain and the resulting advantages will be evaluated.

Curb Mass	m_c	900 [kg]
Passengers + payload	m_p	300 [kg]
Frontal section	A_f	2 [m²]
Drag coefficient	C_x	0.33
Rolling resistance coefficient	C_{RR}	0.016

Table 1.1 Compact electric car design characteristics

Equation 1-1 is used to determine the vehicle's steady state power losses at the target speed. The equation expresses the mechanical power (P_m) required by the vehicle as a function of the speed (v) and of the parameters contained inTable 1.1.

$$P_m = \frac{1}{2}\rho_{air}\, C_x\, A_f\, v^3 + C_{RR}\, g\, m_v\, \cos\alpha\; v + g\, m_v\, \sin\alpha\; v.$$

Equation 1-1 Road Load

Where:

- m_v is the total vehicle mass. It includes the curb mass in march order m_c and the mass of the passenger and the payload m_p;
- ρ_{air} is the air density which is 1.2041 $[kg/m^3]$ at sea level;
- g is the conventional gravitational acceleration equal to 9.80665 $[m/s^2]$;
- α represents the road grade $[rad]$;
- C_{rr} is the tires coefficient of friction;
- The multiplication of C_x (coefficient of aerodynamic friction) by the frontal area of the vehicle A_f results in the aerodynamic drag coefficient.

Figure 1.1 shows the graphical result of the speed-dependant forces acting on the vehicle and the required power.

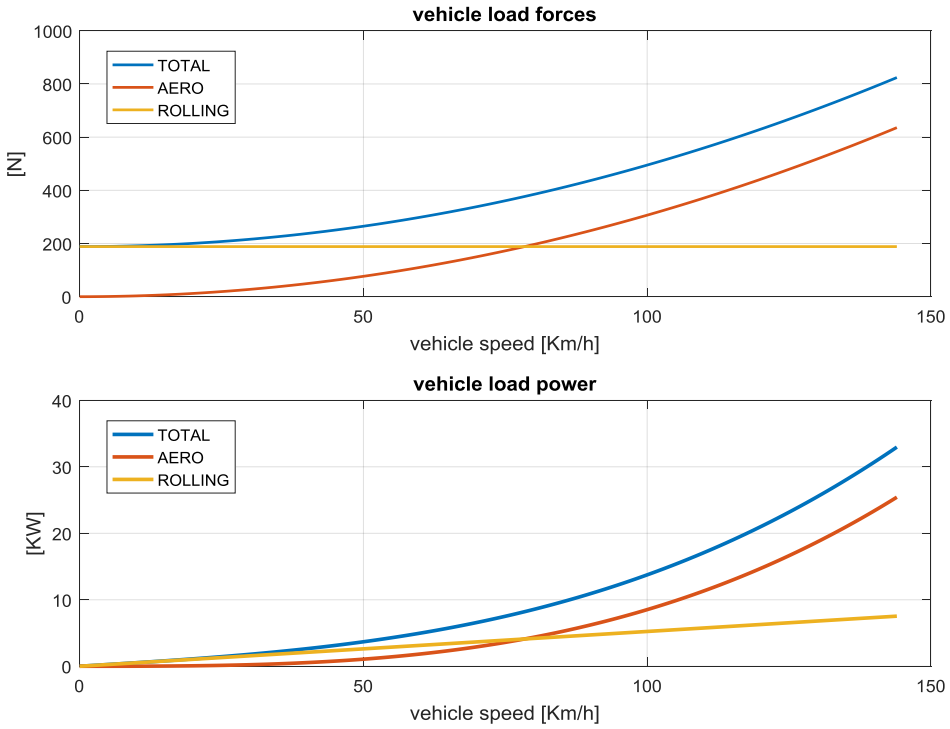


Figure 1.1 Vehicle Road Load

Accounting for the vehicle segment and its performance requirements, the results of the analysis gave a good indication of the total power that could satisfy the target top speed, in this case 30 kW. Therefore, further investigation was carried out to confirm the dynamic behaviour goals using this value.

In order to evaluate the required power related to the vehicle dynamic performance, the simple model above described is applied with several standard cycles and with the chosen available vehicle power. Figure 1.2 and Figure 1.3 show the simulation results related to the *ARTEMIS-Rural* and the *ARTEMIS-Highway* driving cycles, respectively. These tests are standard driving cycles used to perform fuel consumption and emissions tests.

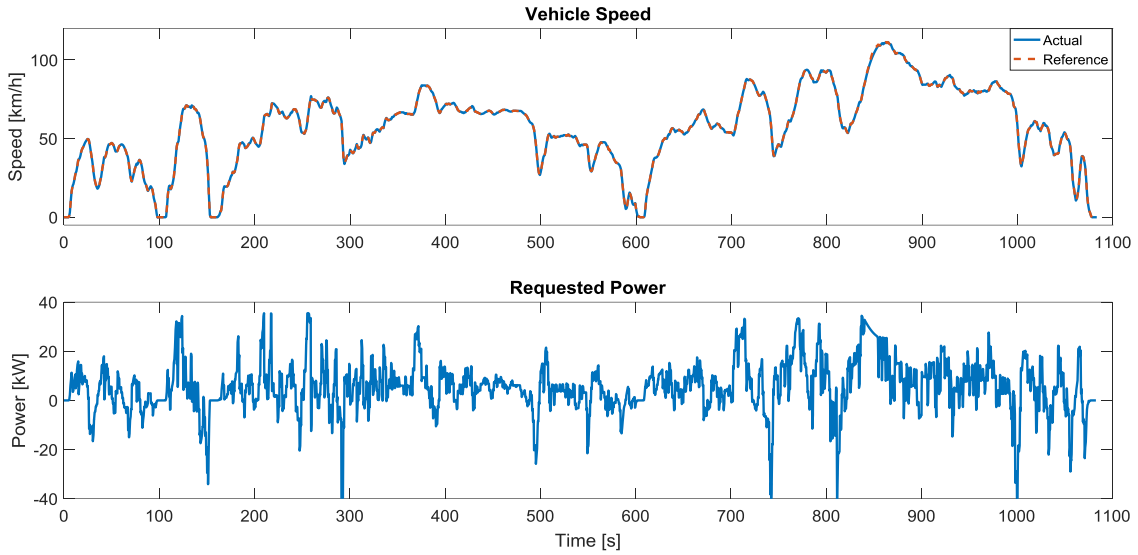


Figure 1.2 ARTEMIS-Rural Simulation

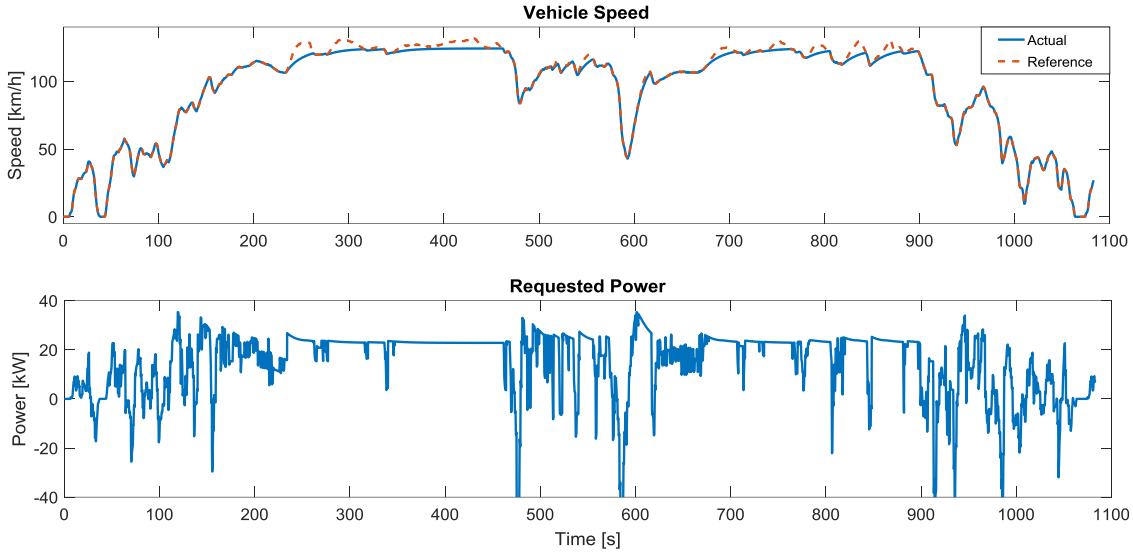


Figure 1.3 ARTEMIS-Highway Simulation

These charts report that, assuming a powertrain power of 30 kW, the vehicle under consideration, having the features indicated in Table 1.1, is able to provide acceptable performance in terms of both acceleration and maximum speed.

1.2 Multi-drive *configurations for electric vehicles*

The possibility to drive a vehicle with more than one motor is widely covered in recent years' literature [4], [5].

Considering two axles vehicles, the most common solutions are those summarized in Figure 1.4 and Figure 1.5. The first shows the "two motors – one driven axle" solutions, where each motor is connected to a wheel of the same axle through a reduction gear.

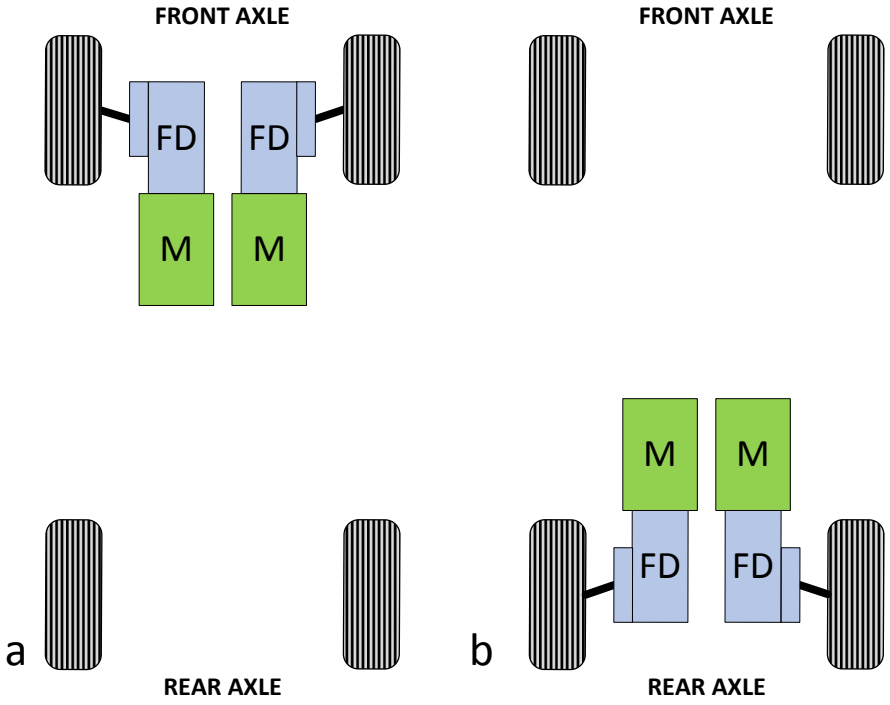


Figure 1.4 Two motors - One driven axle

Solutions with "two motors – two driven axles" (Figure 1.5a) are common as well: the two motors are connected one to each axle through a reducer/differential gear set.

Moreover, literature also considers traction systems with four direct drive motors integrated in the wheel hub, as shown in Figure 1.5b.

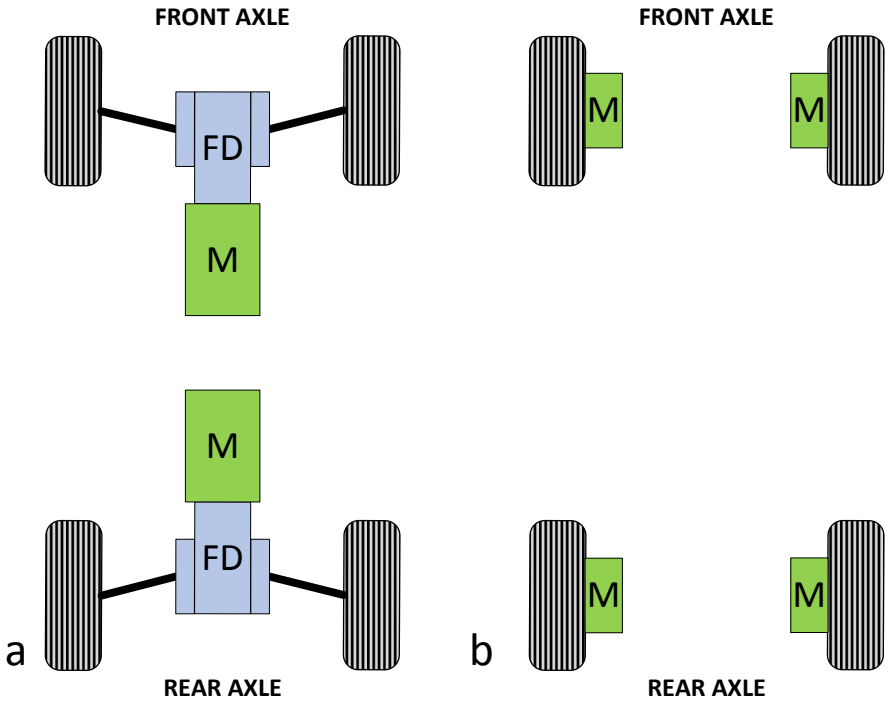


Figure 1.5: a) Two motors- two driven axles; b) Four hub-motors

It is proved that multi-drive powertrains, compared to those that provide only one motor, give some outstanding benefits:

Lower power rating, since the total power is divided among two or more components, these can be designed for a lower rated power, thus being more compact and easier to accommodate in the vehicle.

Higher reliability as the propulsion is allotted to different drives, even in case of failure or malfunction of one of them the vehicle may be able to move, although with reduced power.

Yaw moment control, the dynamics control is a topical issue, especially since it is related to safety aspects of vehicles. One of the main aspects is related to the lateral stability in turns.

Considering the simplified model of Figure 1.6, the cornering vehicle dynamics is defined by the state variables $\dot{\beta}$ (the vehicle slip angle) and $\dot{\gamma}$, namely *Yaw Rate*, that is the body angular speed relative to the z (vertical) axis [6], [7], [8].

Safety standards for cars requires a minimum level of stability performance that can be only satisfied by installing an Electronic Stability Control (ESC). ESCs are very complex systems that estimate the vehicle ideal trajectory from speed, acceleration and steering angle, and, by acting individually on the braking system of each wheel, are able to generate a yaw moment capable to limit the effects of a loss of stability when cornering. ESC reduces oversteering and understeering of the vehicle in a plurality of driving condition improving automobile handling even in case of tire losing grip. Electric vehicle must comply with the same specification for vehicle stability performance of standard cars.

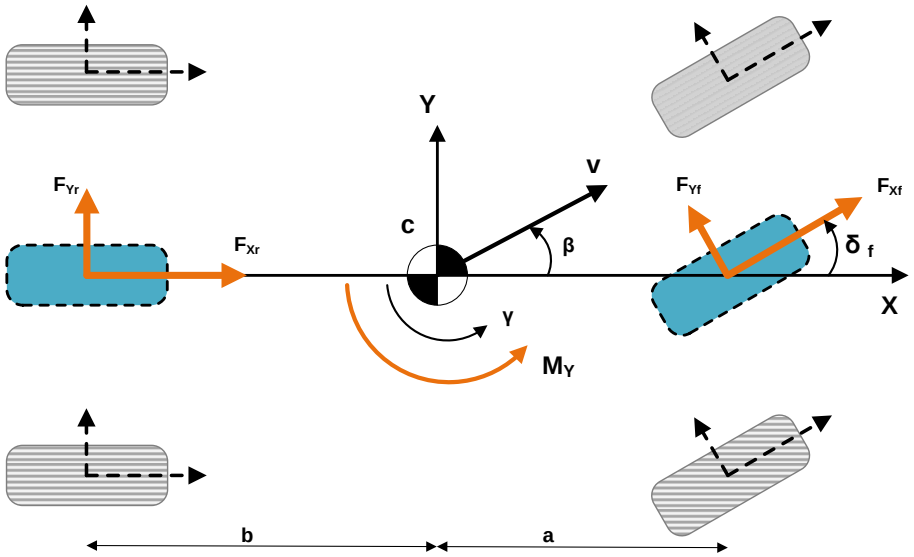


Figure 1.6 Two DOF vehicle model

An electric vehicle with four motorized wheels, as the example of Figure 1.5b, can control the torque of each wheel in both directions, positive and negative, allowing to generate a moment along the *z*-axes of the vehicle [9]. This moment can be used in order to either prevent losing of stability or cooperate with the ESC system, increasing the stability of the vehicle with respect to the sole use of the braking system. This condition is met, although with lower effectiveness, even for the other configurations shown in Figure 1.4 and Figure 1.5 [10].

Extended high power region, taking as an example the "two drives – two driven axles" configuration of Figure 1.5a, the power is split between two different drives. In other words, by employing two motors with the same rated power but with different characteristic or by choosing different final drive ratios for the two motors, it is possible to extend the high-power operating range as in the example of Figure 1.7 [11], with improved performance both at high speed and low speed compared to the single drive configuration with full power.

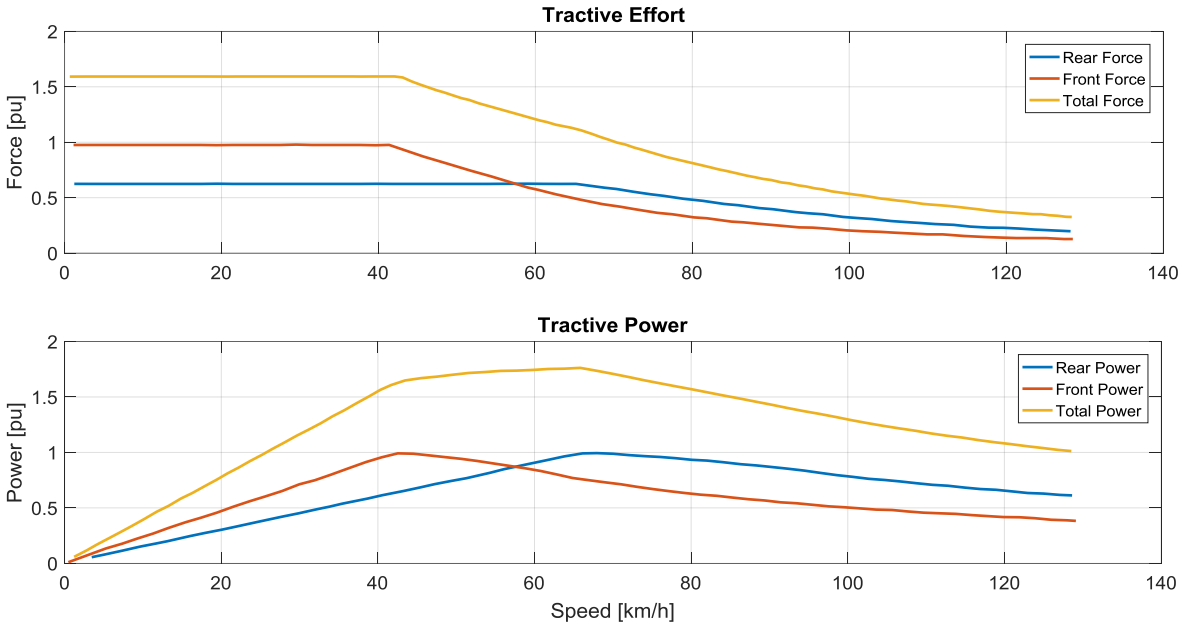


Figure 1.7 Dual-drive force and power characteristics

The benefit is also valid for the other configurations of Figure 1.4 and Figure 1.5, howsoever the Ph.D. activity was performed considering the "two drives – two driven axles" configuration of Figure 1.5a.

The configurations shown in Figure 1.4 and Figure 1.5 have advantages and disadvantages from the point of view of performance and vehicle integration complexity.

The solution that provides for a motor directly coupled to each of the four wheels is certainly the one that allows the best dynamics control capabilities because it can effectively control the force applied to each wheel in both directions. However, this configuration is particularly complex under the design and construction point of view, because the motor must fit in the wheel hub, sharing the limited room with the brake disk, calliper and wheel suspension.

The solutions with two motors driving the two wheels of the same axle have lower efficacy with regard to the vehicle dynamics. Moreover, they show a limit in terms of performance due to the fact that the power is applied only to two wheels, therefore friction is a limiting factor as well.

The "two drives – two driven axles" configuration is also less effective than Figure 1.5b under the stability control potential. Even so, it represents a good trade-off between the presented solutions. This configuration was chosen for the Ph.D. project. The subsequent discourse will refer largely to this type of traction system.

Notwithstanding the several advantages described above, these systems have an additional degree of freedom which is the power sharing between drives, which may be an advantage but it still represents a variable to manage and therefore an increased complexity from the control point of view.

Furthermore, this is a characteristic they have in common with hybrid powertrains, which by definition are *multi-drive* and consisting of at least two energy sources.

The Ph.D. activity also concentrated on the control algorithms development for these powertrains, this topic will be discussed in the following chapters.

Figure 1.8 reports two spread hybrid powertrain configurations for cars: Figure 1.8a is the well-known *Series-Parallel Power-Split Hybrid* configuration, which employs a planetary gear set, an internal combustion engine (ICE) and two electric machines. Figure 1.8b refers to the *Parallel through the road* configuration, where the conventional ICE powertrain and an electrical powertrain are installed on different axles, so that they are not mechanically coupled unless through the road surface.

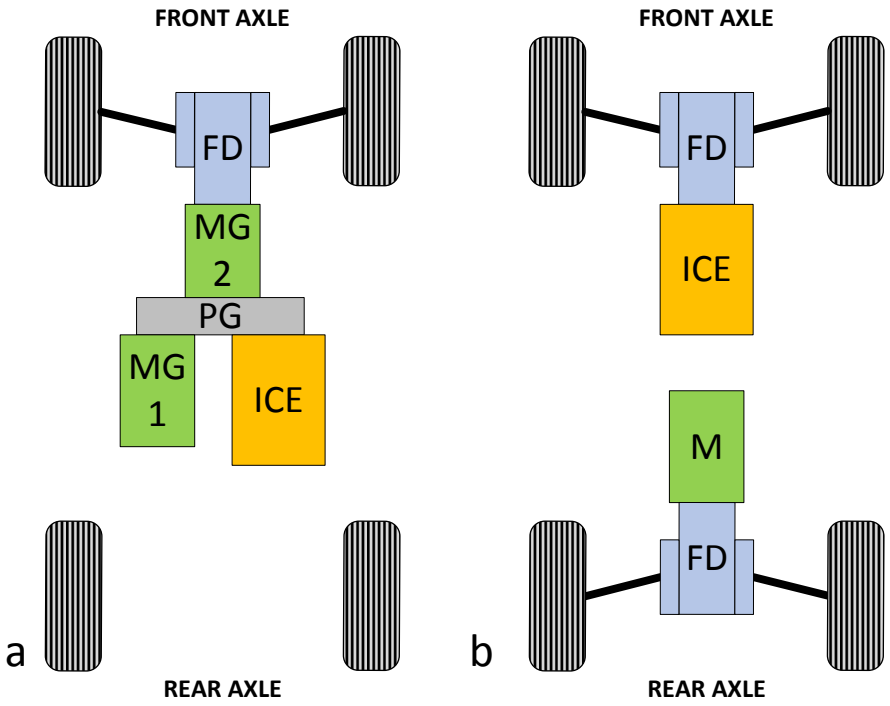


Figure 1.8 a) Series-Parallel power-split hybrid; b) Parallel "through the road" hybrid

1.3 System voltage analysis

A key point for the optimal design of an electric powertrain is the choice of the system voltage (therefore the battery voltage) and the resulting inverter's rated voltage.

The possibility to develop a "two drives – two driven axles" configuration using low and extra-low voltage solutions was investigated within the Ph.D. research activity [12], [13]. A comparison among different available components is presented hereinafter. Figure 1.9 compares the power-cost density of active components suitable to be used for the power stage of a three-phase traction inverter. This preliminary comparison does not take into account the mounting and assembling cost.

As it is widely known, IGBTs are preferred when working at higher voltage, while MOSFETs are preferred at lower voltage. This first analysis indicates that there is a commercial availability of MOSFETs and IGBTs with similar power-cost density and that the current MOSFET technology shows its optimal power-cost density at lower voltage levels. The very high voltage is out of the scope of the project for the induced cost of battery pack, power wiring harness and EM shields and also considering the relatively low power of the system to be developed. The comparison has been focused on MOSFET technology. Figure 1.10 shows the single MOSFET performance in terms of theoretical converted power vs. lost power. Under this point of view, MOSFETs with very similar and very good performance are available for almost all the voltage range.

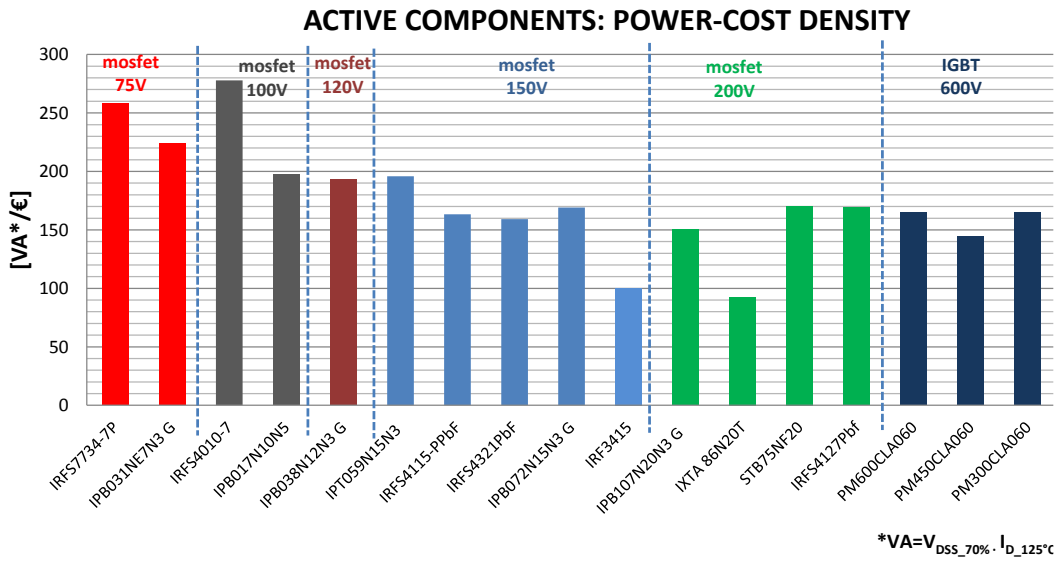


Figure 1.9 Power-cost density for commercially available power active components suitable for the power stage of traction inverter

Considering the single MOSFET analysis, the three-phase inverter is designed assuming a target power output of 15 kVA, according to the simulations described in Paragraph 1.1 and in view of the fact that two drives will be used. The rated battery voltage considered in this analysis ranges between 50 V and 130 V. Table 1.2 summarizes the corresponding electrical characteristic of five possible inverters in rated and overload condition.

For each solution, starting from the MOSFET characteristic, it is possible to choose the right number of MOSFETs to be connected in parallel. An even number of parallel MOSFET is necessary due to layout optimization. Table 1.3 gives the main MOSFET characteristics for every component under study. Table 1.4 shows the number of MOSFETs in parallel required to obtain the performance demanded in Table 1.2 and the resulting real performance of the inverter.

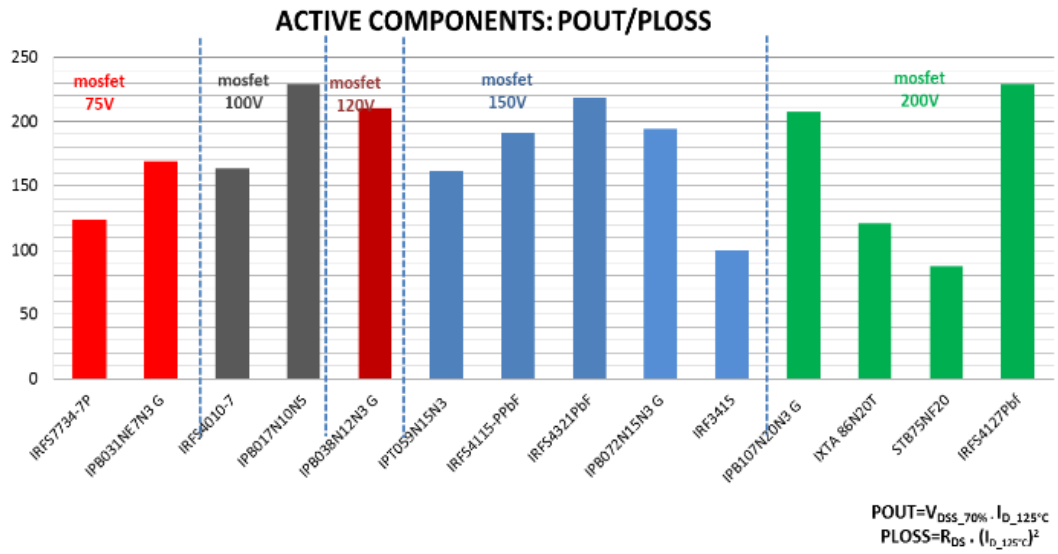


Figure 1.10 Performance factor of commercially available MOSFETS suitable for realizing the power stage of traction inverter

Description	unit	Values				
Component break-down voltage	[V]	75	100	120	150	200
Approx. number of Ion-lithium cells		16	20	24	32	40
Approx. rated DC-link voltage	[V]	52	67	80	105	140
Minimum line-to-line voltage for obtaining the rated motor performance	[V_{RMS}]	33	42	52	67	84
Maximum rated current from the inverter	[A_{RMS}]	260	210	170	130	105
Maximum overload current from the inverter for 240 s	[A_{RMS}]	520	420	340	260	210
Maximum overload current from the inverter for 60 s	[A_{RMS}]	700	560	450	340	280
Maximum battery power	[kW]			27		

Table 1.2 Electrical data of the inverter supply for different DC voltage ratings

n.	Name	Manufacturer	package	I_d 125°C [A]	Breakdown voltage V_{BRDS} [V]	reference battery voltage [V]	R_{DSON} [mΩ]
1	IRFS7734-7P	IR	D2Pack-7pin	139	75	52	3.05
2	IPB031NE7N3 G	INFINEON	D2Pack	100	75	52	3.10
3	IRFS4010-7	IR	D2Pack-7pin	130	100	67	3.30
4	IPB017N10N5	INFINEON	D2Pack-7pin	180	100	67	1.70
5	IPB038N12N3 G	INFINEON	D2Pack	105	120	80	3.80
7	IPT059N15N3	INFINEON	HSOF-8	110	150	105	5.90
8	IRFS4115-PPbF	IR	D2Pack	55	150	105	10.00
9	IRFS4321PbF	IR	D2Pack	40	150	105	12.00
10	IPB072N15N3 G	INFINEON	D2Pack	75	150	105	7.20
11	IRF3415	IR	D2Pack	25	150	105	42.00
12	IPB107N20N3 G	INFINEON	D2Pack	63	200	140	10.70
13	IXTA 86N20T	IXIS	D2Pack	40	200	140	29.00
14	STB75NF20	ST	D2Pack	47	200	140	34.00
15	IRFS4127Pbf	IR	D2Pack	51	200	140	12.00

Table 1.3 Main characteristics of the considered active components (MOSFETs)

n.	Name	V_{BRDS} [V]	MOSFET in parallel	Equivalent R_{DS} [mΩ]	Output rated current [A_{RMS}]	Output maximum current [A_{RMS}]	Nominal power [VA]	Maximum power [VA]
1	IRFS7734-7P	75	6	0.51	237	710	15240	45721
2	IPB031NE7N3 G	75	8	0.39	227	681	14619	43857
3	IRFS4010-7	100	4	0.83	148	443	12670	38009
4	IPB017N10N5	100	4	0.43	204	613	17543	52628
5	IPB038N12N3 G	120	6	0.63	179	536	18420	55260
7	IPT059N15N3	150	4	1.48	125	374	16081	48243
8	IRFS4115-PPbF	150	8	1.25	125	374	16081	48243
9	IRFS4321PbF	150	12	1.00	136	409	17543	52628
10	IPB072N15N3 G	150	6	1.20	128	383	16446	49339
11	IRF3415	150	16	2.63	113	340	14619	43857
12	IPB107N20N3 G	200	6	1.78	107	322	18420	55260
13	IXTA 86N20T	200	8	3.63	91	272	15594	46781
14	STB75NF20	200	6	5.67	80	240	13742	41226
15	IRFS4127Pbf	200	6	2.00	87	260	14911	44734

Table 1.4 Main parameters and performance for possible inverter configuration

For any inverter configuration, Figure 1.11 and Figure 1.12 represent conduction power losses for output powers of 15 kVA (rated condition) and 35 kVA (overload condition) respectively.

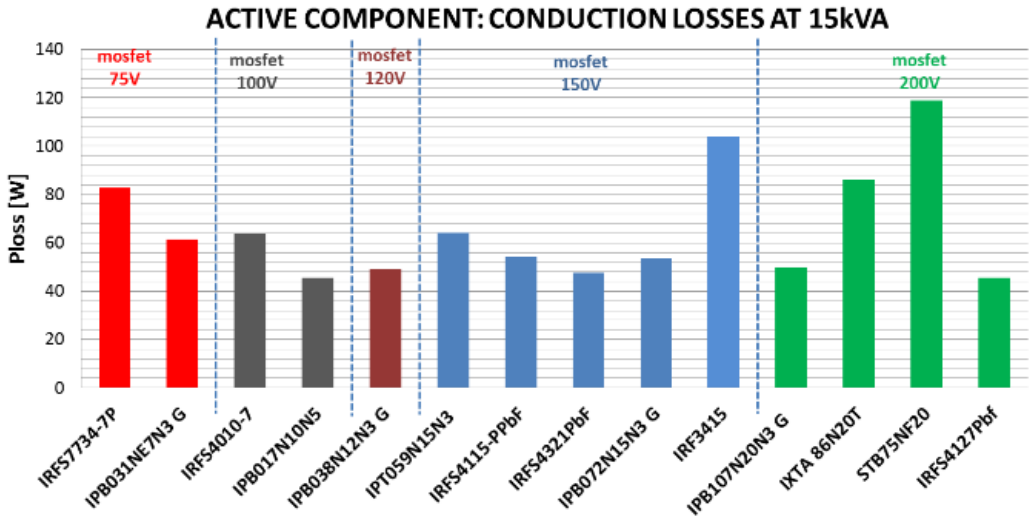


Figure 1.11 Conduction losses of three-phase traction inverters realized with different MOSFETs at 15kVA (rated condition)

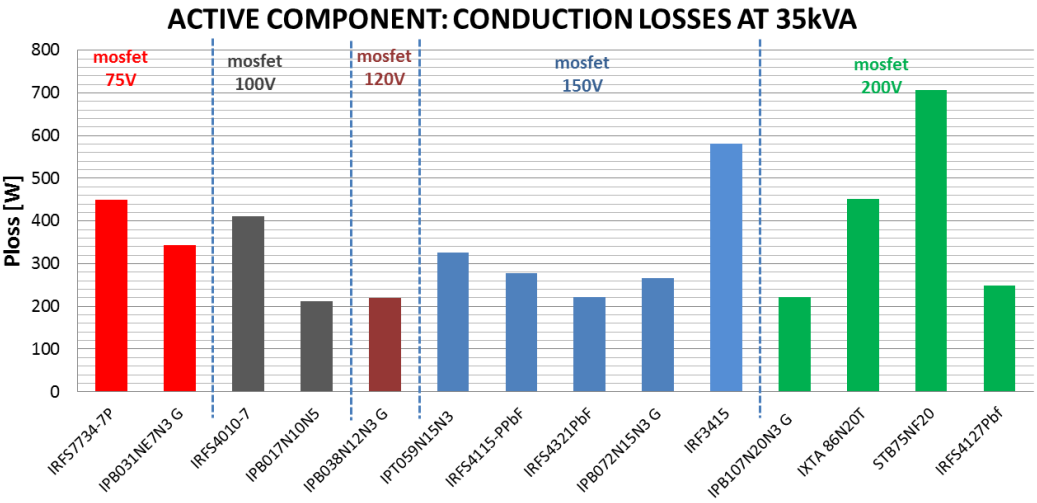


Figure 1.12 Conduction losses of three-phase traction inverters realized with different MOSFETs at 35kVA (overload condition)

The cost of the active components roughly represents 30% to 50% of the cost of the power stage of the inverter. The higher the current, the higher the cost of the power circuit and then the lower the percentage of the active components cost. Recent improvement in mounting technologies (IMS, DBC and Thick copper PCB) contribute to lower the cost of the power circuitry even for very high current output.

The cost of the DC-link capacitors is mainly associated to the “kVA” rating, but a certain dependency can be found to the DC-link voltage. For example, the cost for the 200 V voltage rating is increased due to lack of suppliers. For this reason, inverter cost comparison is performed taking into account only the active components. Resulting components cost for each inverter is given in Figure 1.13.

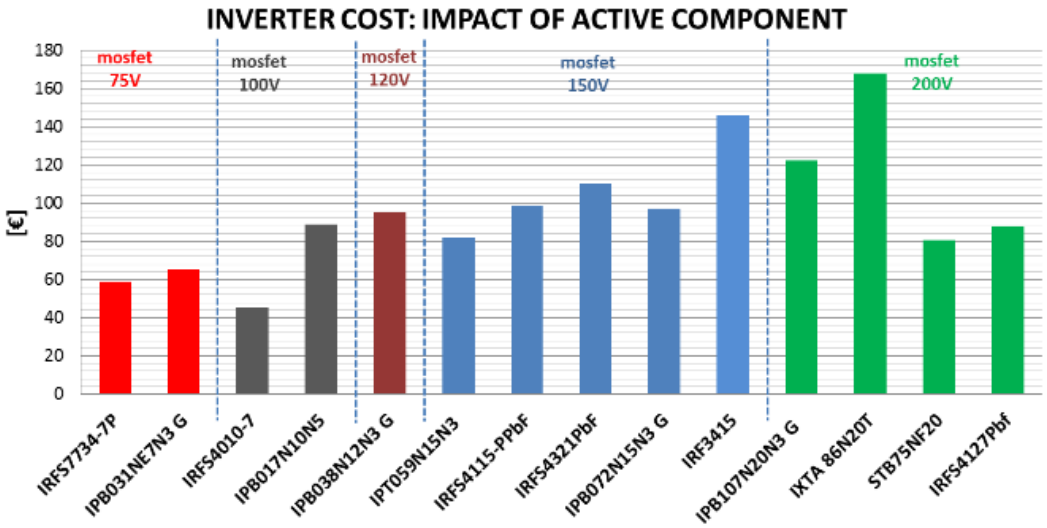


Figure 1.13 Component costs for a 15kVA three-phase traction inverter with different MOSFETS

The results obtained with the different solutions can be summarized as follows:

- All the voltage range (V_{BRDS}) between 75 V and 200 V can be used for the selected application. It means a battery pack composed of 16 to 42 cells connected in series.
- The number of MOSFET in parallel ranges from 4 to 16. A number higher than 12 should be avoided in order to minimize the complexity of the power circuits. The higher the number of power MOSFETs in parallel, the wider the planar footprint of the inverter power stage.
- Voltage level V_{BRDS}=120V has only one supplier. For this reason, at the time of this research, it should be avoided.
- Losses are minimized for the 100 V and 150 V rating. The lower the conduction losses, the simpler the design of the cooling system.
- Voltage level V_{BRDS}=100 V has the best results in terms of cost. It is worth noting that the best solution at V_{BRDS}=150 V has a cost of active component 60% higher than the best solution at V_{BRDS}=100 V.

A first conclusion of this analysis is that the MOSFETs in the voltage classes V_{BRDS}=150 V and 100 V represent the two best options, at the time of this research, for realizing an automotive inverter with a rated power of approximately 15 kVA and a maximum power of at most 35 kVA.

A second conclusion is the demonstration of the technical and economic feasibility of very low voltage system for the power range under investigation.

From this analysis, the design of the two traction inverters using MOSFETs at V_{BRDS}=75 V for the DC-link voltage in the range 48-53 V (approximatively 14 to 16 ion-lithium cells in series) is fully justified [11].

Chapter 2
Traction management algorithm

The vehicle's powertrain configuration, as discussed in Chapter 1, is of the type "two drives – two driven axles". Additionally, each drive is powered by a separate battery pack, as shown in Figure 2-1. The two drives are therefore independent of each other. However, their high-level control is centralized.

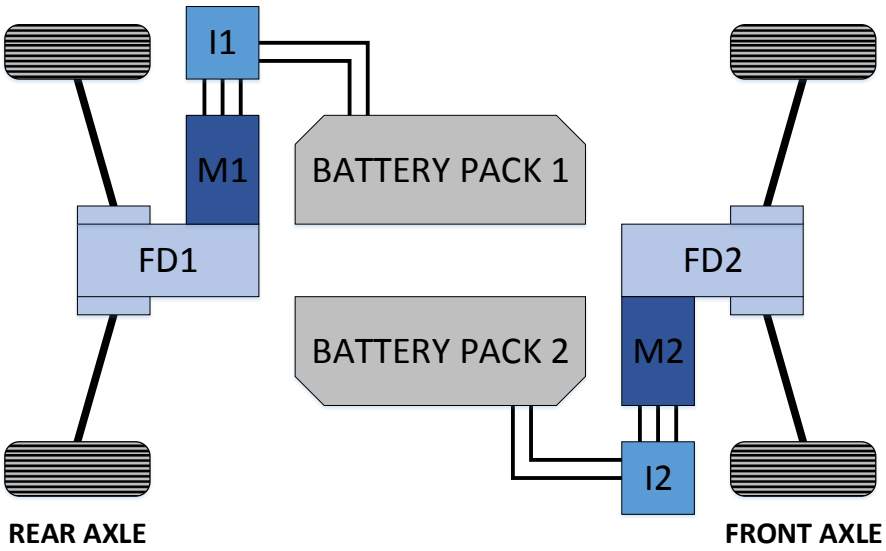


Figure 2-1 Traction system layout

For the described powertrain configuration, the aim of the high-level control (also referred to as *Traction Management*) is to generate a torque reference for both the front and rear drives, taking into account the driver's demand as well as the actual system state and the commands coming from an external stability control algorithm, which takes the priority and can modify the absolute output reference. It is worth noting that the traction management and the stability control were developed independently in two different environments and a great deal of work was carried out by the candidate to put them together on the vehicle hardware.

The control algorithm described in this chapter is developed in the MATLAB/Simulink environment with the possibility, as it will be described afterwards, to reuse the simulation model, along with proper compiling tools, to automatically generate executable code for the vehicle ECU. For the correctness of the model configuration, some aspects were considered and described in this chapter.

Figure 2-2 gives an overview of the traction management model. The algorithm receives inputs (green blocks) from the field (measurements, status signals, etc.) and from the driver (commands). It computes all of them outputting the torque reference for both rear- and front-drive, making use of predefined maps, complying with the protection limitations and giving the possibility to receive an external priority reference from a stability control algorithm.

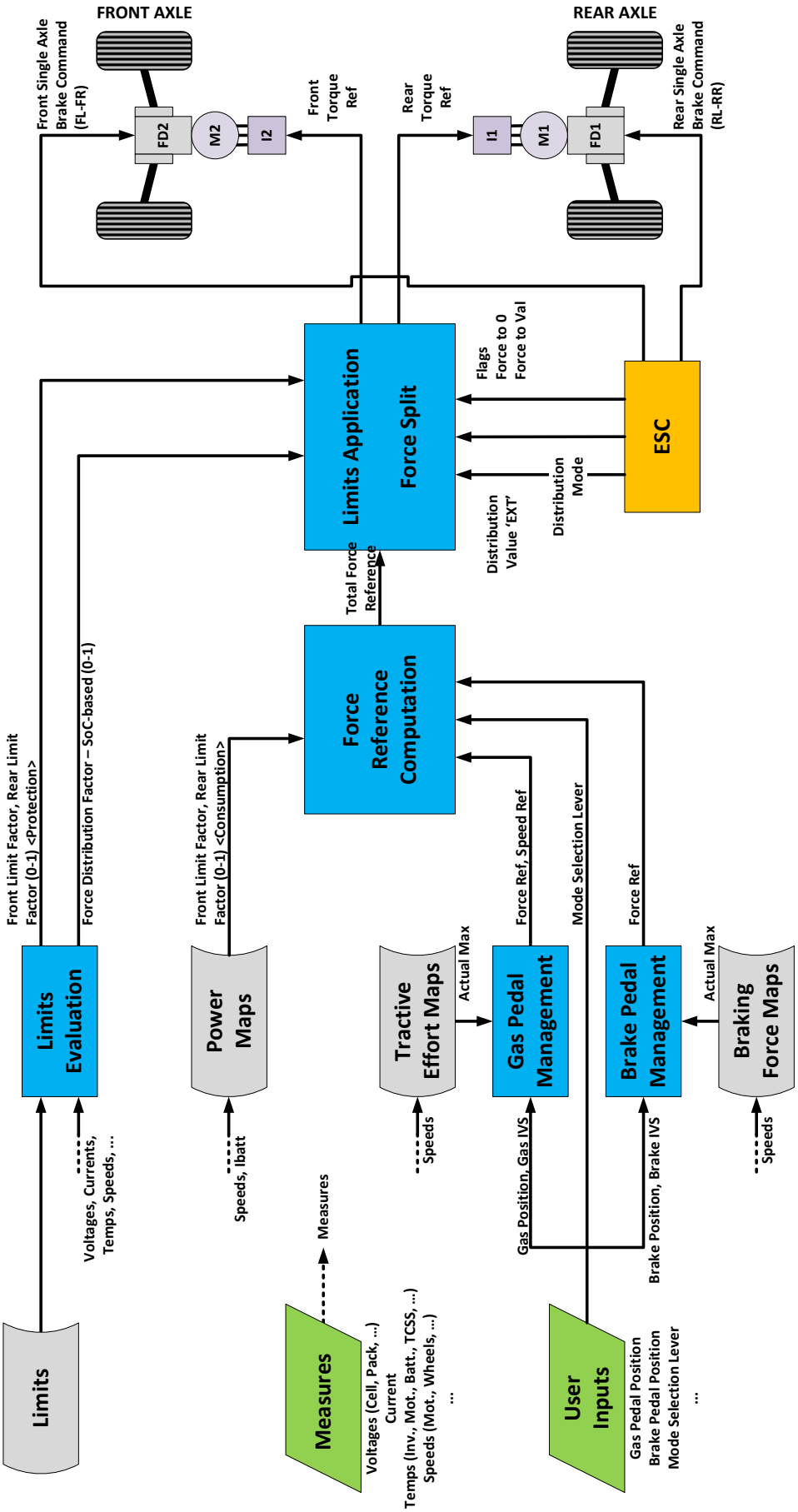


Figure 2-2 Traction Management Scheme

In the following, the blocks of Figure 2-2, which correspond to the main functions, are described.

2.1 Gas Pedal Management, Brake Pedal Management

The *Pedal Management* functions are in charge of computing a vehicle force reference, according to the driver inputs and force maps.

Figure 2-3 and Figure 2-4 show the actual Simulink block diagram of the *Gas* and *Brake* pedal respectively.

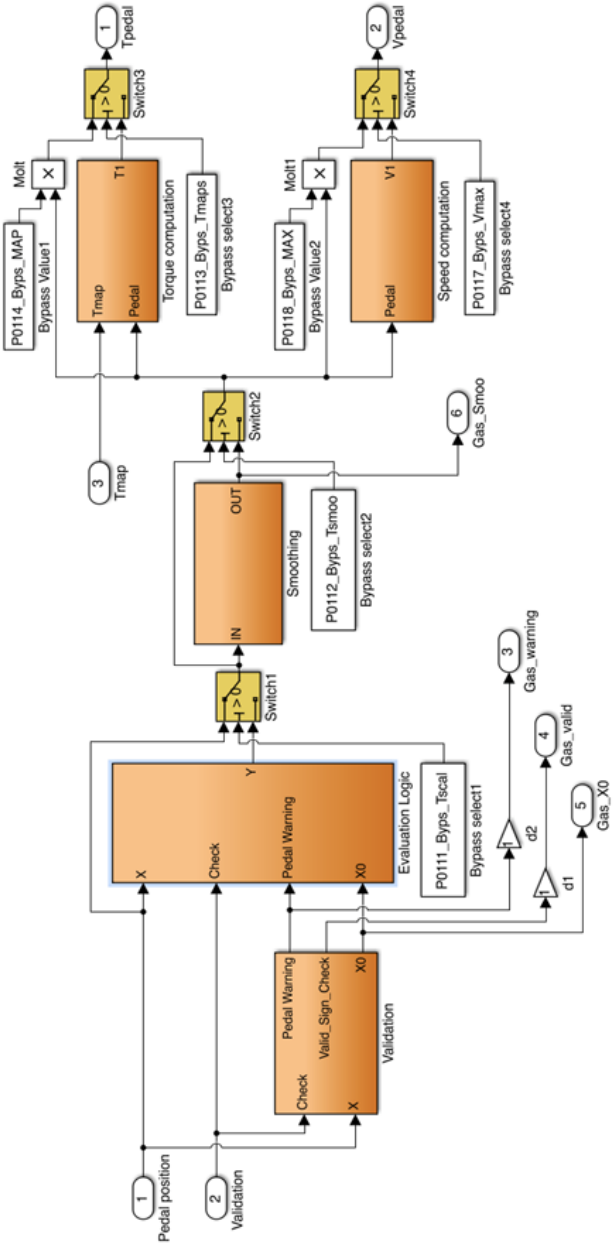


Figure 2-3 Gas Pedal Management block diagram

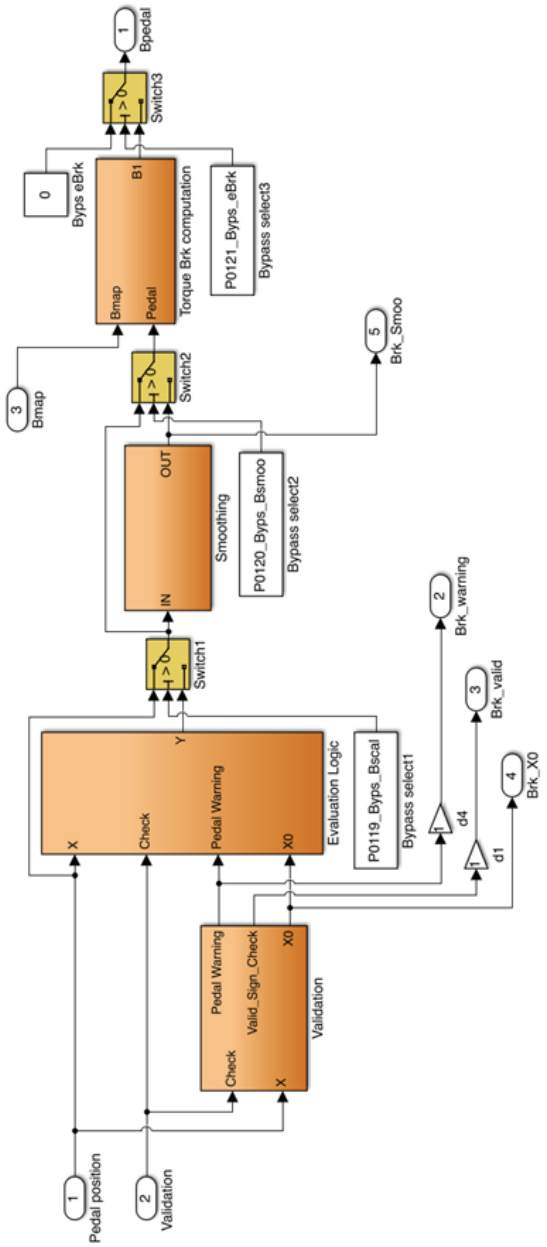


Figure 2-4 Brake Pedal Management block diagram

The information coming from the pedal position sensor consists on an analog signal, proportional to the actual stroke, and a digital *idle validation signal* "IVS", which is used as a safety feature: the IVS signal is 0 when the pedal is completely below a given threshold, usually 10% of the full pedal travel, and it turns into 1 if the stroke exceeds the threshold.

The input blocks check the correct behaviour of the pedal sensor by monitoring these signals. The position value is then rescaled, according to a parameterized rule, into a *per-unit* "pu" which streamlines the subsequent operations. An example of this operation is given in Figure 2-5.

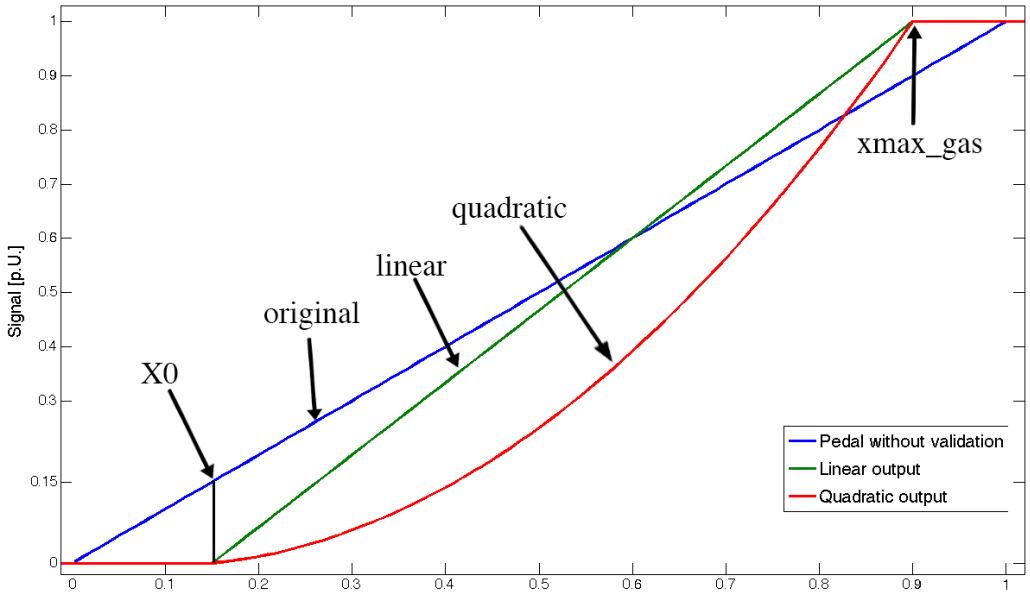


Figure 2-5 Pedal rescaling and linearization

The *pu* signal then undergoes to a smoothing process, which eliminates dangerous discontinuities in the signal that would translate into peaks of requested torque, harmful for the drivetrain and inconvenient for the driving experience.

The conditioned pedal position signal is finally converted into a force reference according to *Tractive-* and *Braking-force Maps.*

These maps are defined in a *Force vs. Speed* reference plane, an example is shown in Figure 2-6. It can be noticed how two maps are defined for the Gas pedal computation, which represent the upper limit and the lower limit, while only the upper limit map is defined for the brake pedal computation, whose lower limit is always 0.

In the force computation, the *pu* input signal is made to cover all the values in between the upper and lower limits, as exemplified in Figure 2-7.

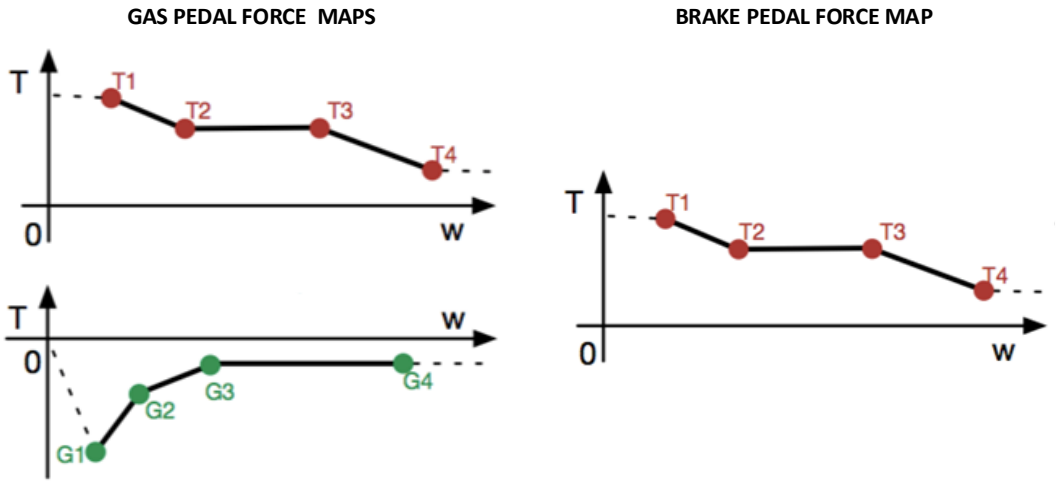


Figure 2-6 Gas- and Brake- force maps

Figure 2-7 shows a typical output of the Gas Pedal Management. The blue line represents the pre-processed gas pedal position, after the validation, ramps and smoothing computation. Cyan and green lines are the mapped force limits, while the red line is the computation of the pedal position force request according to these limits.

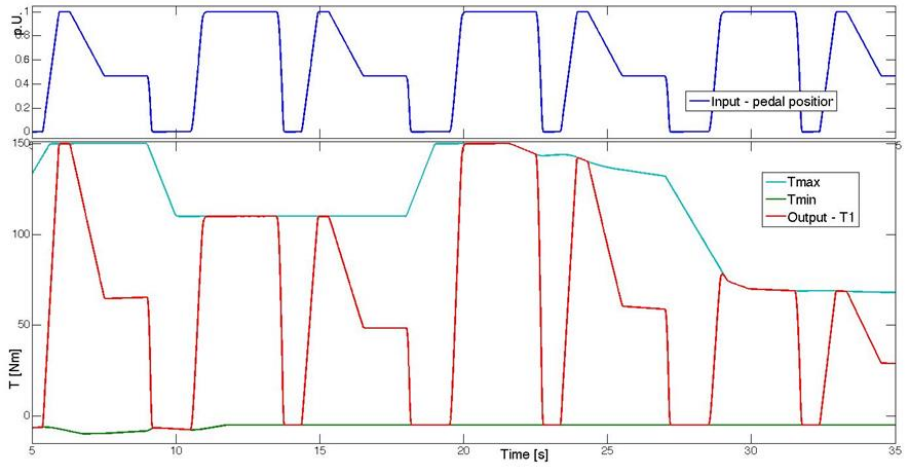


Figure 2-7 Tractive-force computation

Different force maps are available for both the Gas- and Brake-pedal Management functions and, since the driver is able to switch the selected map at run-time, a smoothing switch routine is implemented, not to deliver a reference step as output during a map change.

The Gas Pedal Management function also implements a speed loop, which improves the driving feeling while manoeuvring at very low speed. The contribution of the speed controller vanishes above a pre-set speed threshold.

2.2 Limits Evaluation

The block “Limit Evaluation” generates two limiting variables, one for each drive, from several monitored system variables (temperatures, speed, voltages, etc.). This function has the main objective to limit each drive to reach protection value of critical powertrain parameters.

The block diagram of Figure 2-8 provides a graphical explanation of the limitation working principle. It refers to the computation of limits for one of the two drives (the rear-drive). An identical one is implemented for the front drive.

The graph shown in Figure 2-9 provides an example of the limiting function intervention. The monitored variable is the rear inverter temperature, for which the limitation is effected proportionally to the temperature if it exceeds 60 °C, up to a total power cut-off the if the temperature reaches 70 °C. The middle part of the graph shows the two limiting coefficients calculated for the front- and the rear-drive, and it shows the limitation coefficient trend compared with increasing temperature. The effect of the limitation on the torque demand is shown in the lower part of the graph.

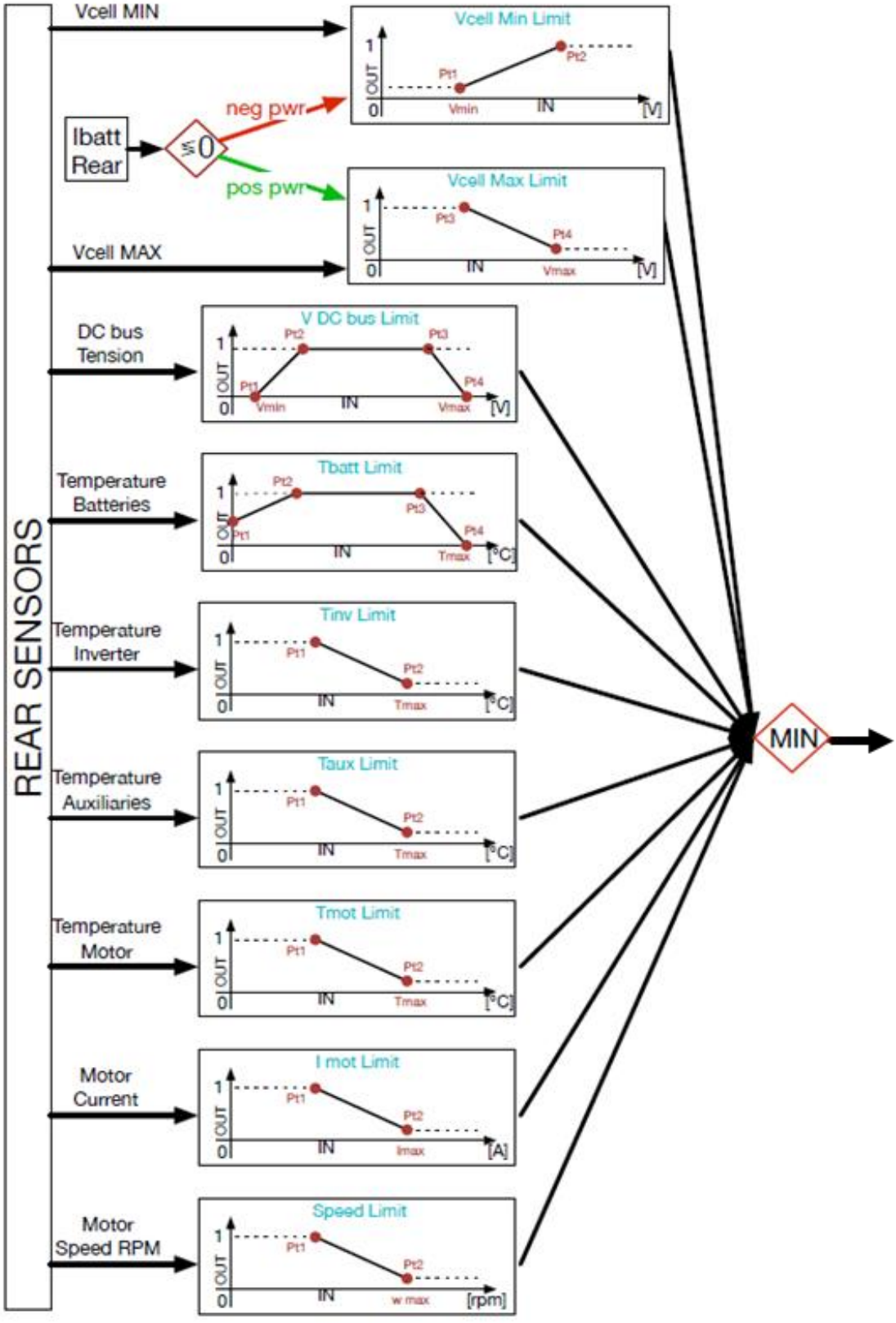


Figure 2-8 Limits Evaluation (Rear)

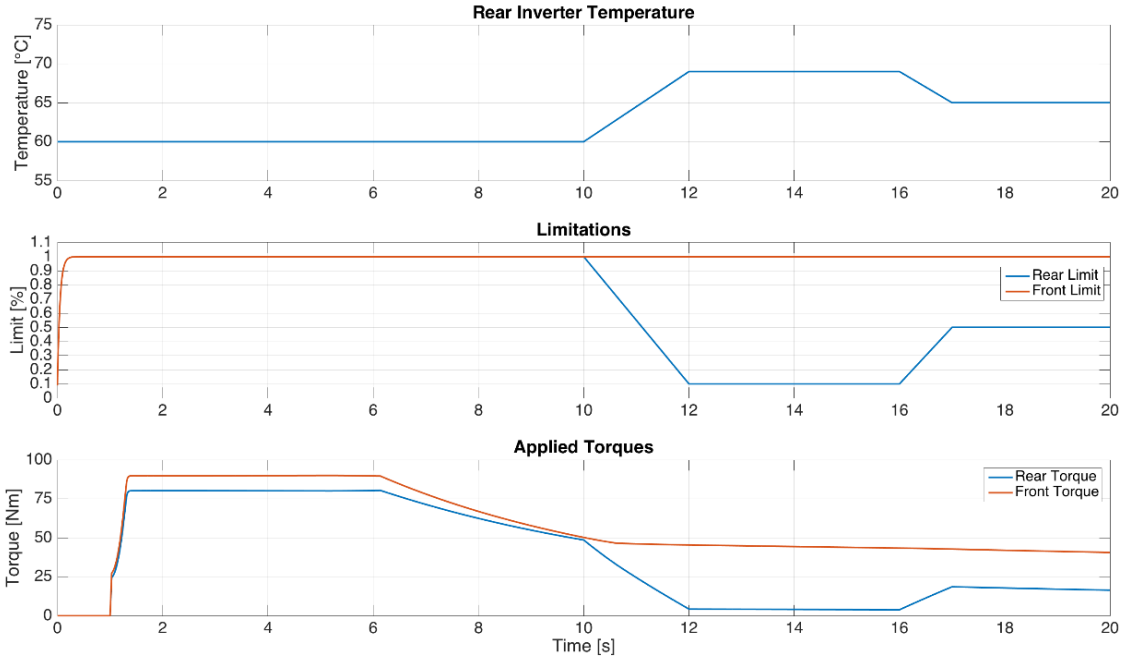


Figure 2-9 Example of limit intervention

2.3 Force Reference Computation

The block represented in Figure 2-2 as *Force Reference Computation* recalculates the total force reference, coming from the Pedal Management, by applying a power limitation function, which in turn makes use of “Power Maps”, ensuring that the power exchanged with the batteries does not exceed predetermined values, defined by the maps.

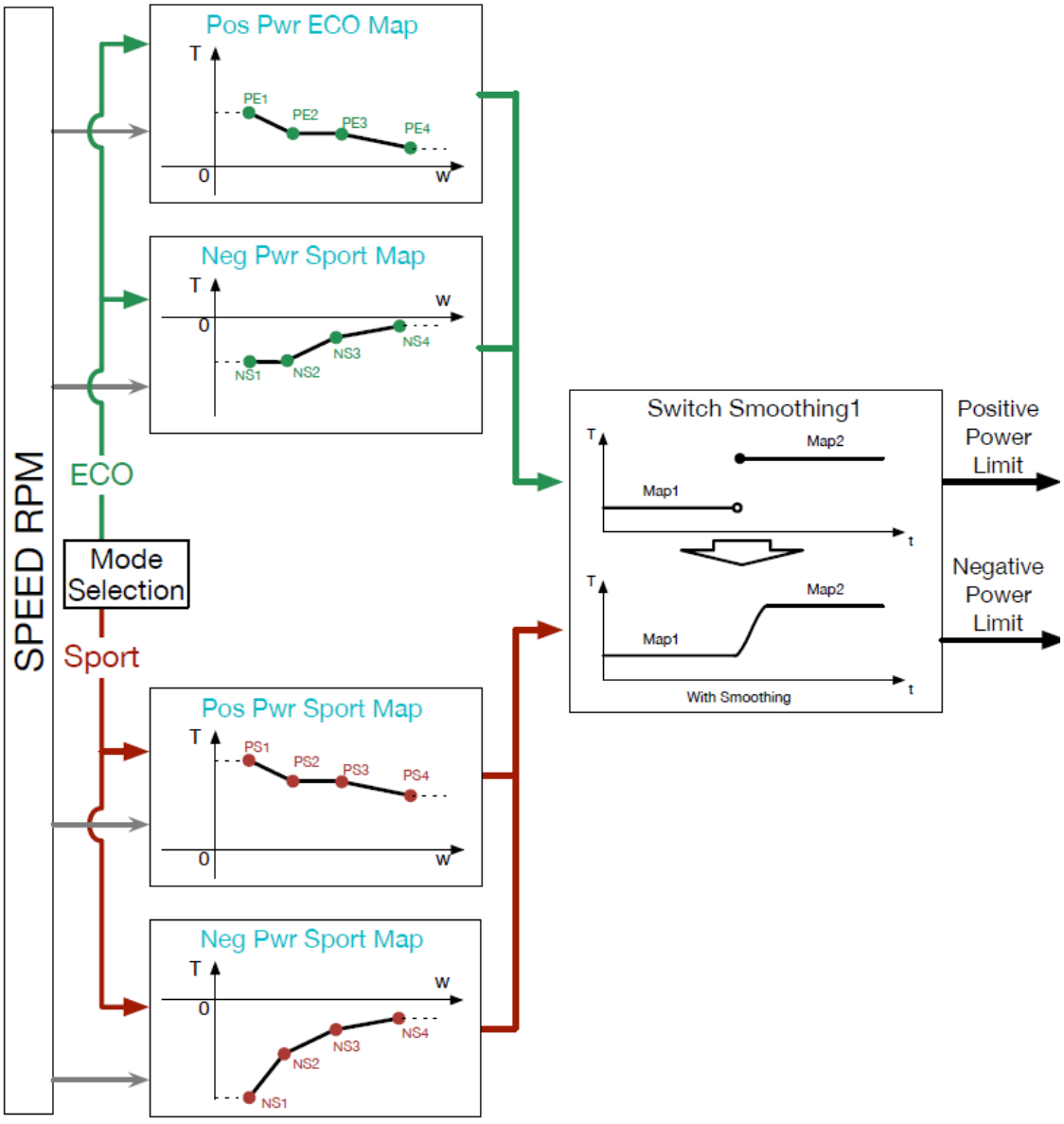


Figure 2-10 Power map selection

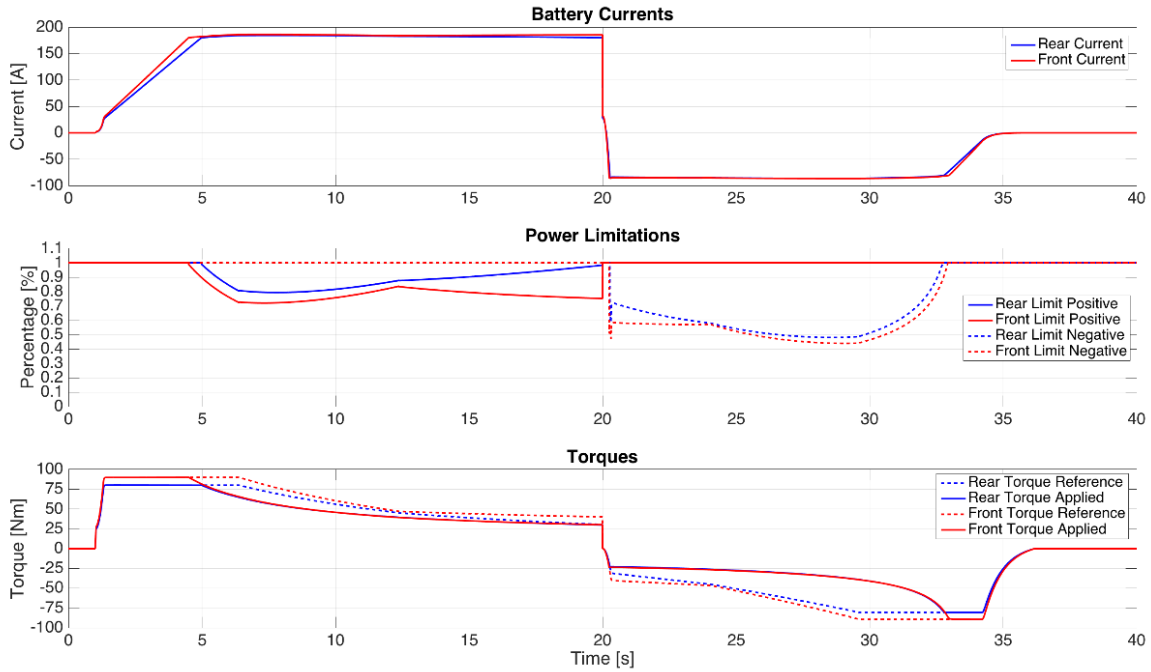


Figure 2-11 Example of map intervention

In this block, the brake reference priority is also managed. It means that an action on the brake pedal always overrides a gas command, thus reducing the risk of mistaken command from the driver.

Moreover, it implements the "gearbox" function, reversing the force reference if the driver puts the direction command in the position corresponding to the backward driving mode.

2.4 Limits Application & Force split

The total demanded force is finally processed in *Limits Application & Force split*. It calculates the demanded force for the front and rear drive by applying a force sharing coefficient ("*split*" coefficient) and the limit coefficient coming from the previous blocks. For each drive, the conversion from force to torque reference is also performed by taking into account the vehicle parameters (transmission/differential ratio and wheel radius).

Moreover, this block performs the key task of applying the commands coming from the Electronic Stability Control algorithm (yellow block of Figure 2-2). In case of stability control intervention, its reference takes the priority and the one coming from the upstream functions is momentarily ignored. The ESC can set to zero one or both torque references. Additionally, a fixed torque reference defined by a parameter, can be applied to one or both drives in order to generate a yaw moment (Paragraph 1.2).

The split coefficient can be provided by the ESC algorithm as well as applied through a fixed value. If the battery energy balance function is enabled, the coefficient may be varied of a small percentage through a variable calculated within the *Limits Evaluation* block, in order to preserve the balance of State of Charge (*SoC*) between the two battery packs.

If a limitation occurs, the force split can be adjusted according to four force-sharing strategies that can be selected by the ESC, or by other high level vehicle management functions.

The four functions are here summarized:

- Mode 1, *Balance preservation*: This mode applies the more stringent limitation on the overall force and then applies the requested split ratio as is.
- Mode 2, *Individual Max*: This mode applies the limitation after the force split. It allows the maximum percentage of available force, for both drives.
- Mode 3, *Total Max Search*: This mode finds the closest level of force to the requested one. If one drive has limitation applied, the other will compensate.
- Mode 4, *Delta Preservation*: This mode preserves the difference in torque values, between front and rear references, even if a limit is applied.

The working principle of the above described block can be graphically summarized through the scheme of Figure 2-12. A simulation is also displayed in Figure 2-12.

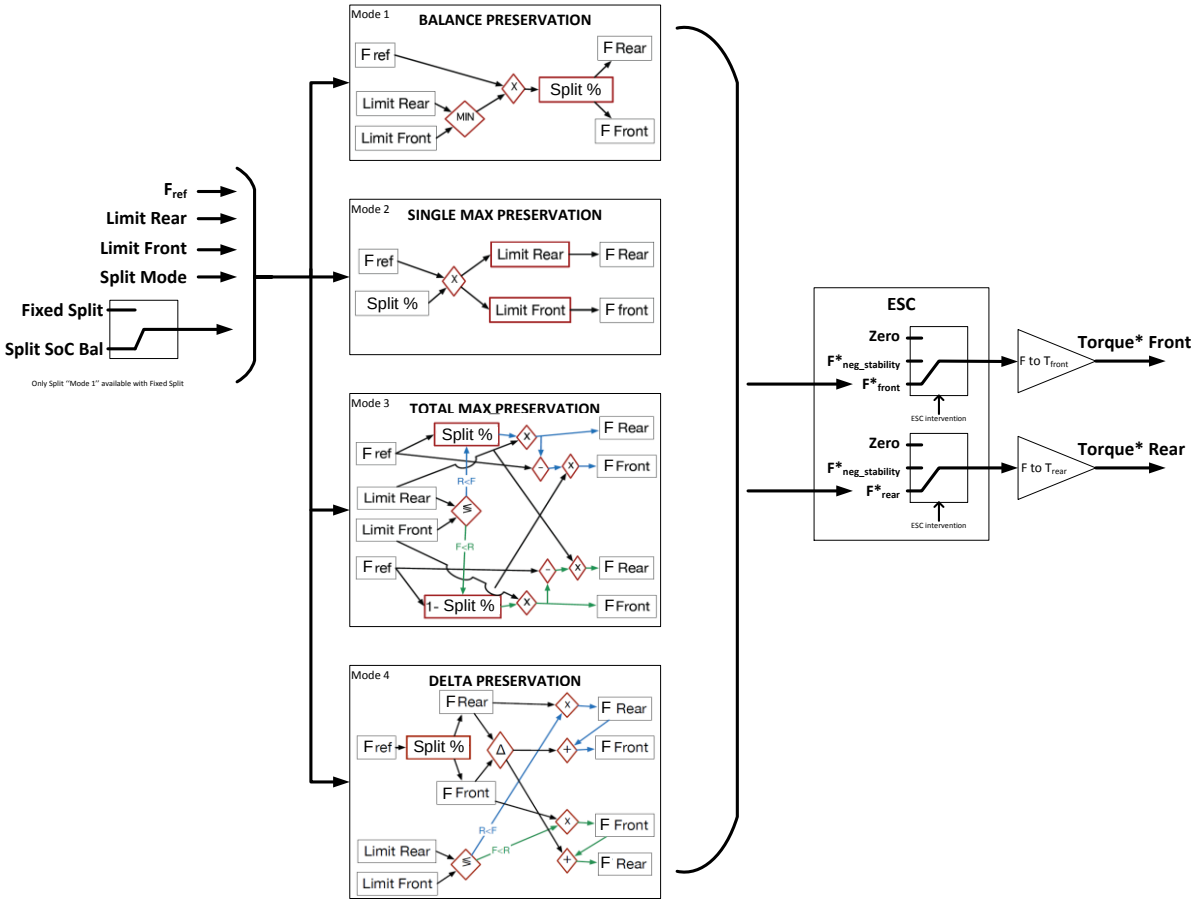


Figure 2-12 Limits Application & Force Split block scheme

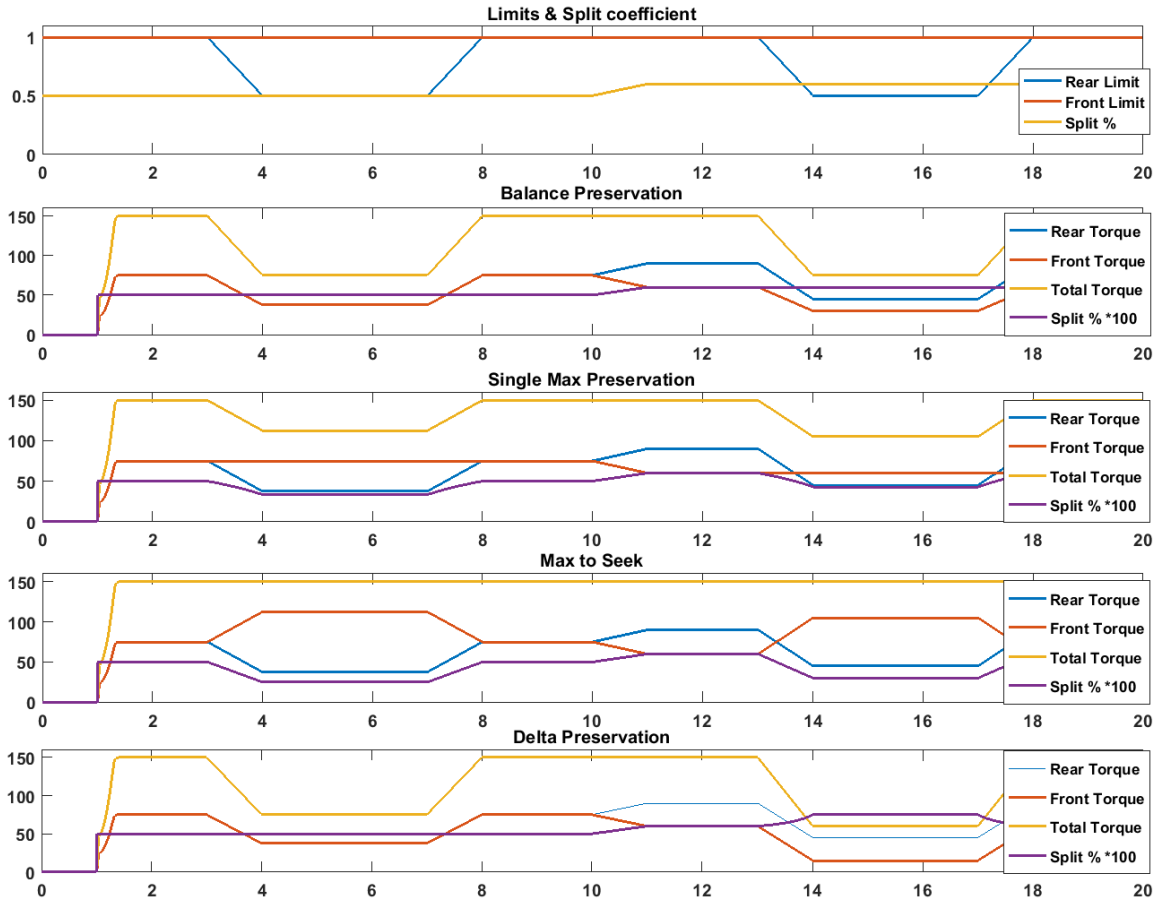


Figure 2-13 Limit and splitting modes simulations

The actual Simulink model of the Traction Management algorithm is shown in Figure 2-14.

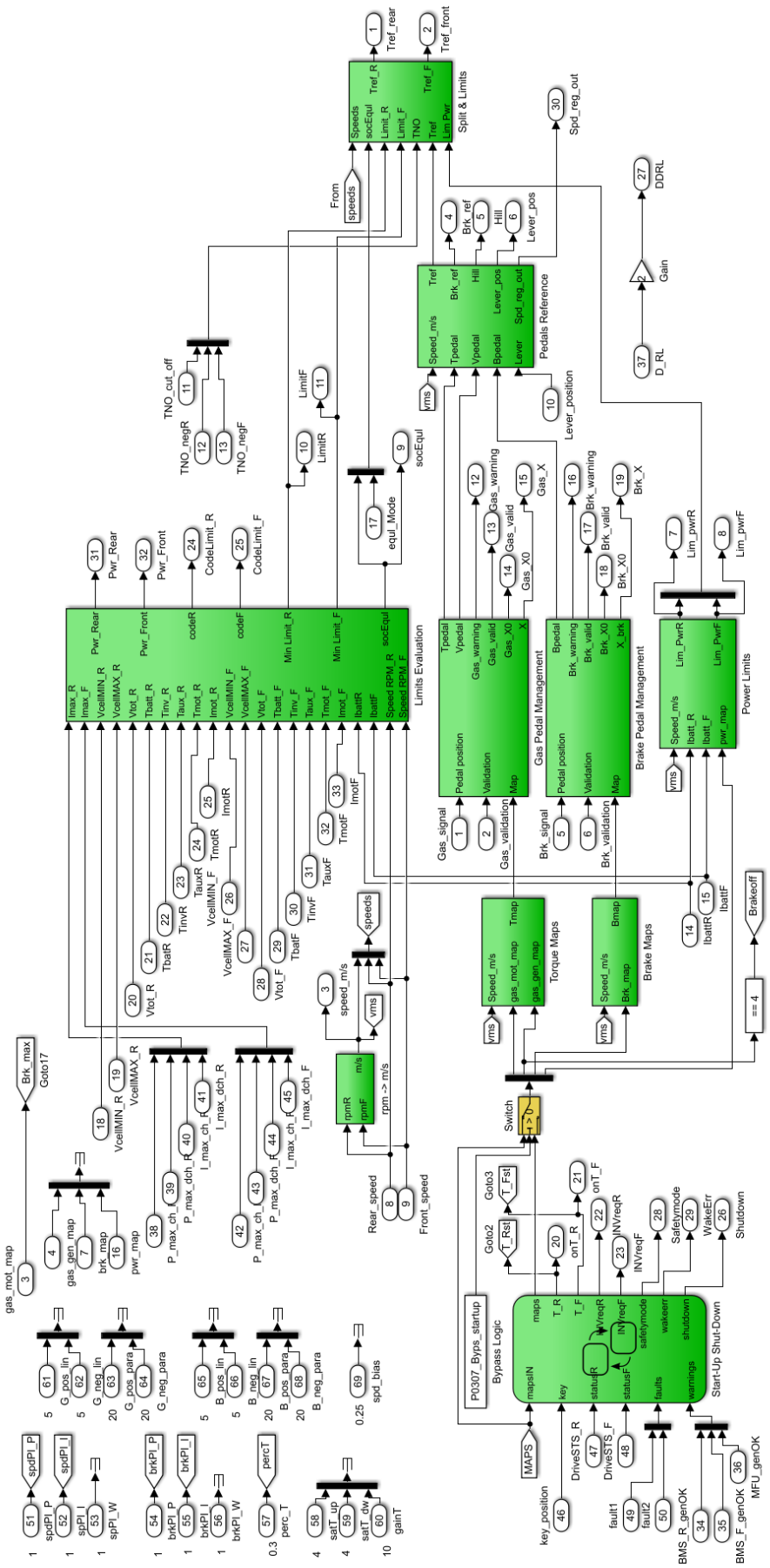


Figure 2-14 Simulink model of the "Traction Management" algorithm

2.5 Additional Logic

Besides the features described above, it contains a *Stateflow* state machine (enlarged in Figure 2-15) for the management of events such as the start up and shutdown sequences and the fault occurrences.

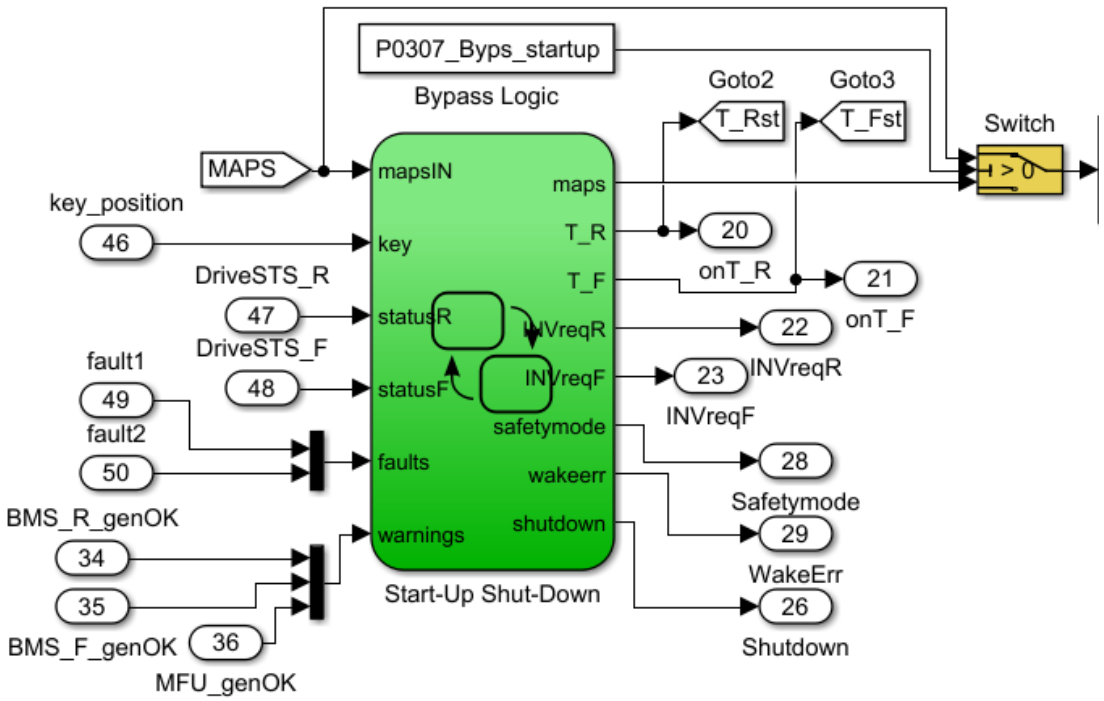


Figure 2-15 State Machine

The aim of the state machine is to prevent undesired behaviour during transient conditions like the vehicle start up and shutdown. In these phases, the torque reference is forced to zero.

At start up, the powertrains are not activated unless the vehicle is turned on with both pedals released and the direction lever positioned at “NEUTRAL”. After the “IGNITION” command, the system waits for the drives to be ready and checks the BMS status. This check is continuously performed during normal operation as well.

The state transition logic is schematized in Figure 2-16.

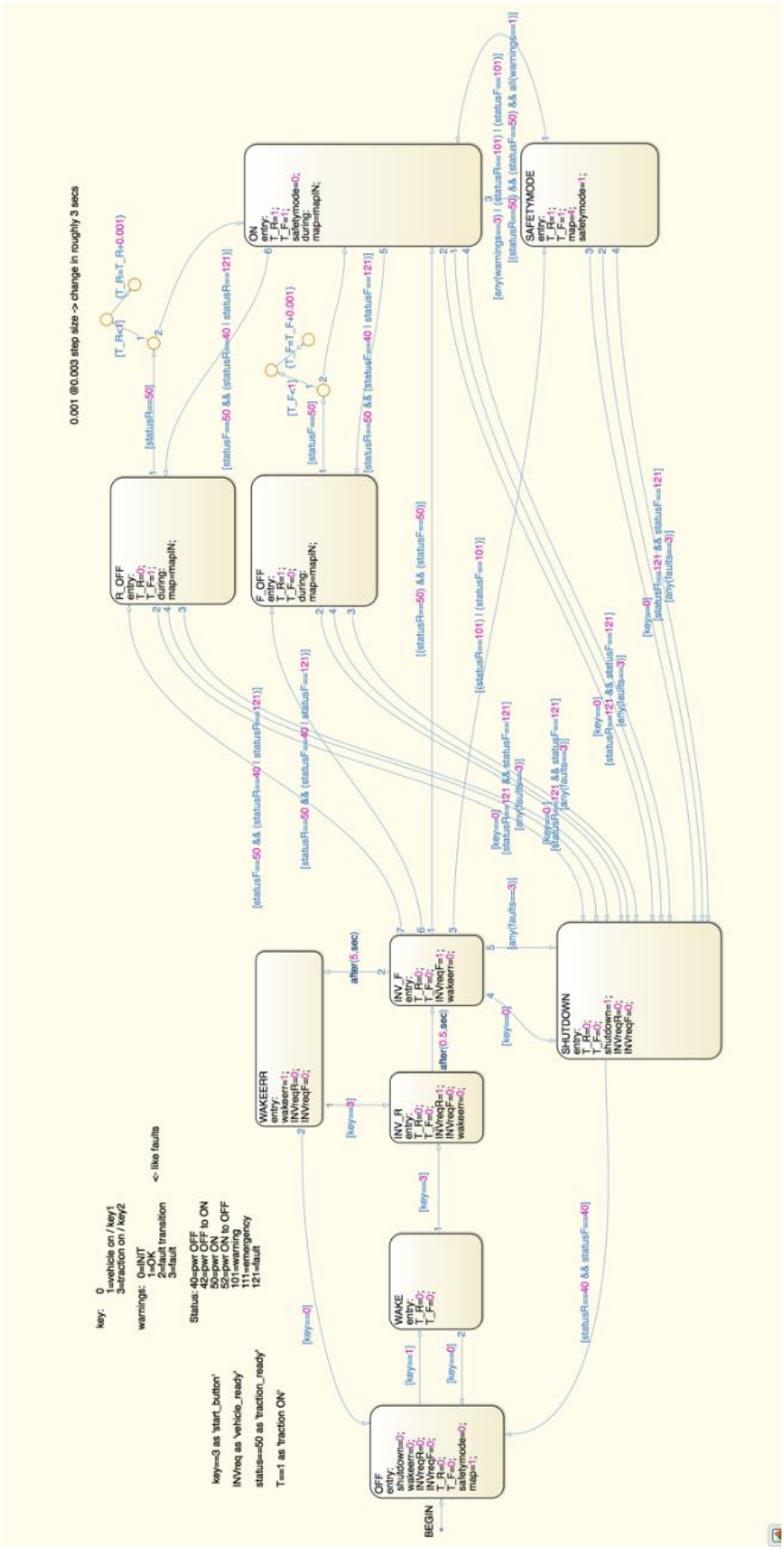


Figure 2-16 State transition logic

As just anticipated, this block is responsible for the implementation of a start-up procedure for the traction control system. It takes into account the key position, both inverter statuses and eventual warnings or errors. The goal of the system is to safely enable traction if no warnings or faults are present in the system.

The power-on procedure requires a predefined sequence of key inputs that enables the inverter power delivery if they are ready and no faults in the systems are detected.

Shut-down relays on the key position but also on emergency situations defined by the inverter status and external fault requests.

Normal start-up procedure is as follows:

1. OFF state:
   This state is encountered at first switch on of the VCU or after SHUTDOWN, the state machine initializes all the outputs in the off condition waiting for the key input to power on.

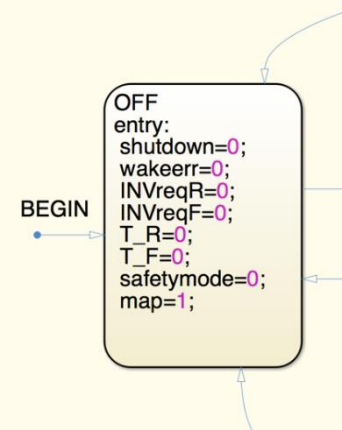


Figure 2-17 "OFF" State

2. WAKE state:
   The driver turns the key into position 1 and the state changes to WAKE. It waits the start up of devices other than the inverters. In this version, it does not introduce any functionality. All the outputs keep the same value.

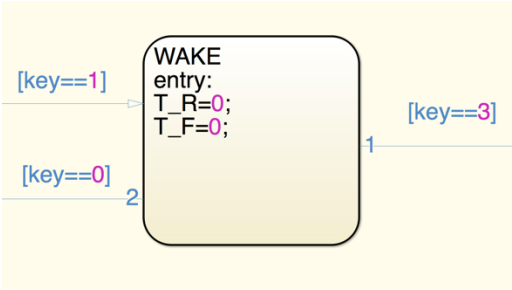


Figure 2-18 "WAKE" State

3. INV_R and INV_F state:
   These two blocks are activated after the driver requests the switch on of the power inverters with the key “IGNITION” (or start button). It implements the correct procedure to power up safely both inverters. The blocks have the same goal, the first one starts the rear inverter while the second one starts the front one. A delay is introduced to avoid the simultaneous closing of the power circuit breakers.

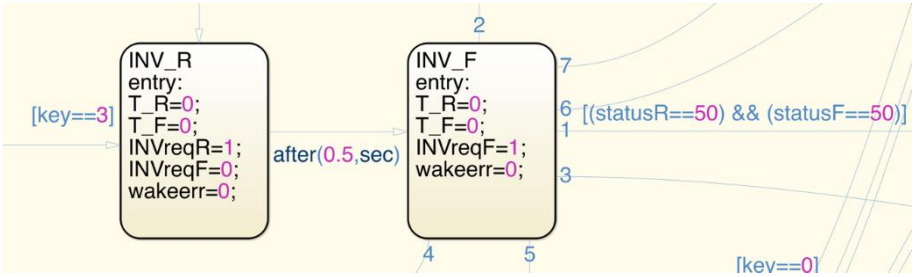


Figure 2-19 "INV_R" and “INV_F” (traction drive) State

Each block produces the output command for the inverter power-on (“*INVreqR*” and “*INVreqF*” respectively). After the command, the two inverters controls are activated, the power stage is energized and, after the procedure is completed and they are ready, they produce the “*Power-ON*” message (status = 50).

4. ON state:
   In this stage the program enables the delivery of the torque request from the traction control system to the inverters by changing “*T_R*” and “*T_F*” to 1.

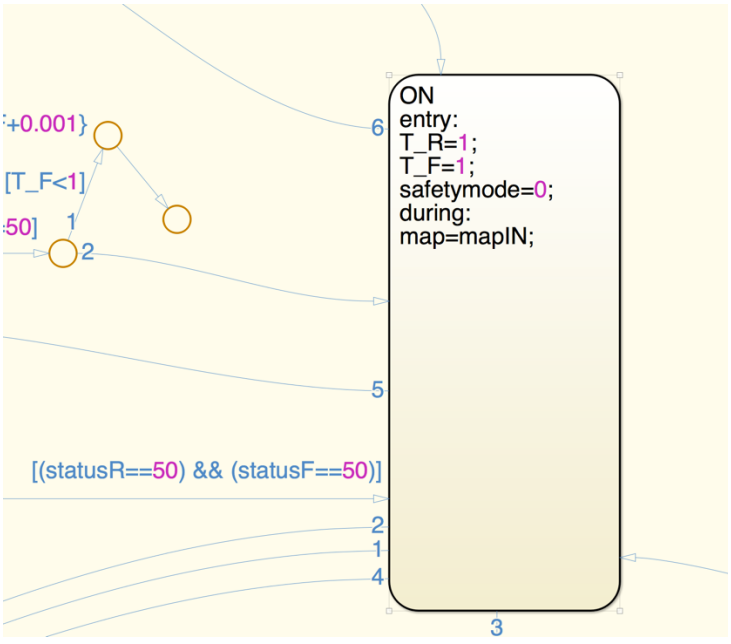


Figure 2-20 "ON" State

During this state, the map selection is left up to the driver.

Now it will be described the normal shut-down procedure from the ON stage:

1. Normal shut down procedure is activated if at least one of these 3 conditions is true:
   a. Key is turned into position “0” (Vehicle OFF);
   b. The inverter statuses are both in the OFF situation;
   c. The system receives a fault message either from the inverters or an external source.
2. SHUTDOWN state:
   If any of the above conditions is met, then the system goes into “*SHUTDOWN*” state. In it, the torque request delivery is turned off by setting to 0 the “T_R” and “T_F” outputs and the power down procedure of the inverters is initialized by putting the outputs “INVreqR” and “INVreqF” to 0. The flag output “*shutdown*” is changed to 1 until the inverter procedure has come to the end. When the inverters are powered down, the system returns to the OFF state.

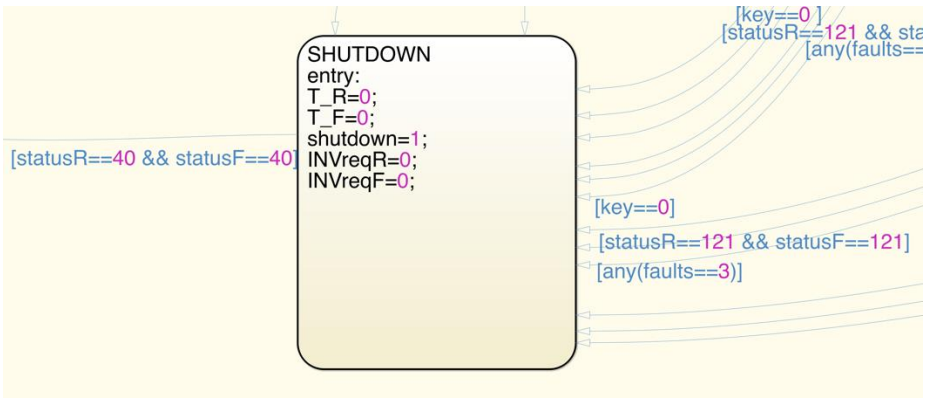


Figure 2-21 “SHUTDOWN” State

Other states are added to the scheme to take into account abnormal situations. They are described below.

- WAKEERR state:
  It is applied if both inverters fail to complete the power-on procedure in 5 seconds. Consequently, the system stops sending the “INVreqR” and “INVreqF” signals and the flag output “wakeerr” is set to 1 to highlight the problem. From this state, key position 2 (IGNITION) is requested to restart the inverters power-on. Otherwise, position 0 puts the system in the OFF state.

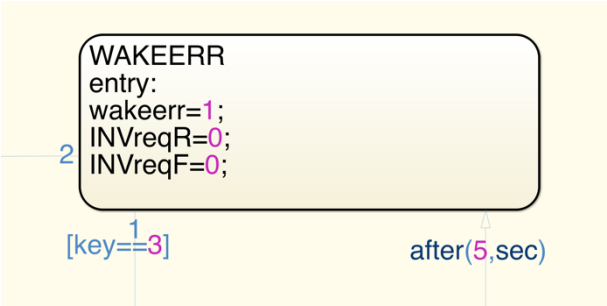


Figure 2-22 "WAKEERR" State

- R_OFF and F_OFF states:
  They are triggered when one inverter completes successfully the power-on procedure but the other does not, either because there is an error or because it takes more than 5 seconds to turn on. This state may also be triggered if a single inverter stops working during normal conditions. In these states the torque reference request is active only for the working inverter using the "T_R" or "T_F" output. Map selection is also free. The inverter power-on requests are left active for the whole duration of this state. When the system receives the input that the other inverter has completed the power-on sequence and it is ready, it transitions to the ON state applying a linear ramp on the appropriate "T_R" or "T_F". Shut-down procedure may apply with the usual conditions.

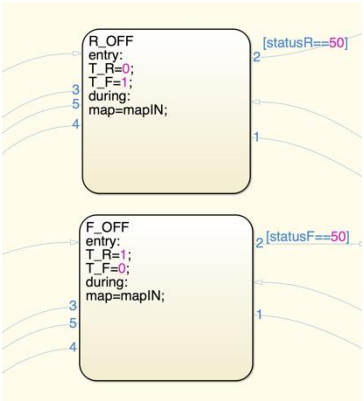


Figure 2-23 "R_OFF" and "F_OFF" States

- SAFETYMODE state:
  It is activated when there is a warning from the inverter's status or from external sources. In this state the torque delivery ("T_R" and "T_F") is enabled, but the torque map 4 is forced on the system. Map 4 being the more restricting one, thus allowing the vehicle to move, but with limited performances. If the warning input ceases to be a problem, normal operation is restored by going to the ON state. Shut down is possible through the normal conditions and procedure.

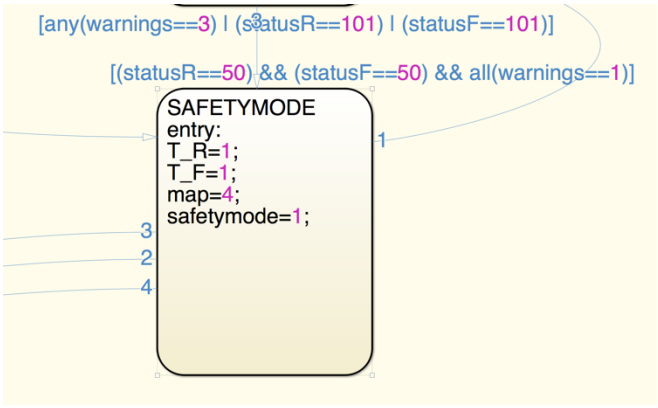


Figure 2-24 "SAFETYMODE" State

The whole Traction Management algorithm is contained in a single block, presented in Figure 2-25.

This representation, with the input ports on the left side of the system block, and the output ports on the right side, is useful for the subsequent automatic code generation, as described in the next chapter.

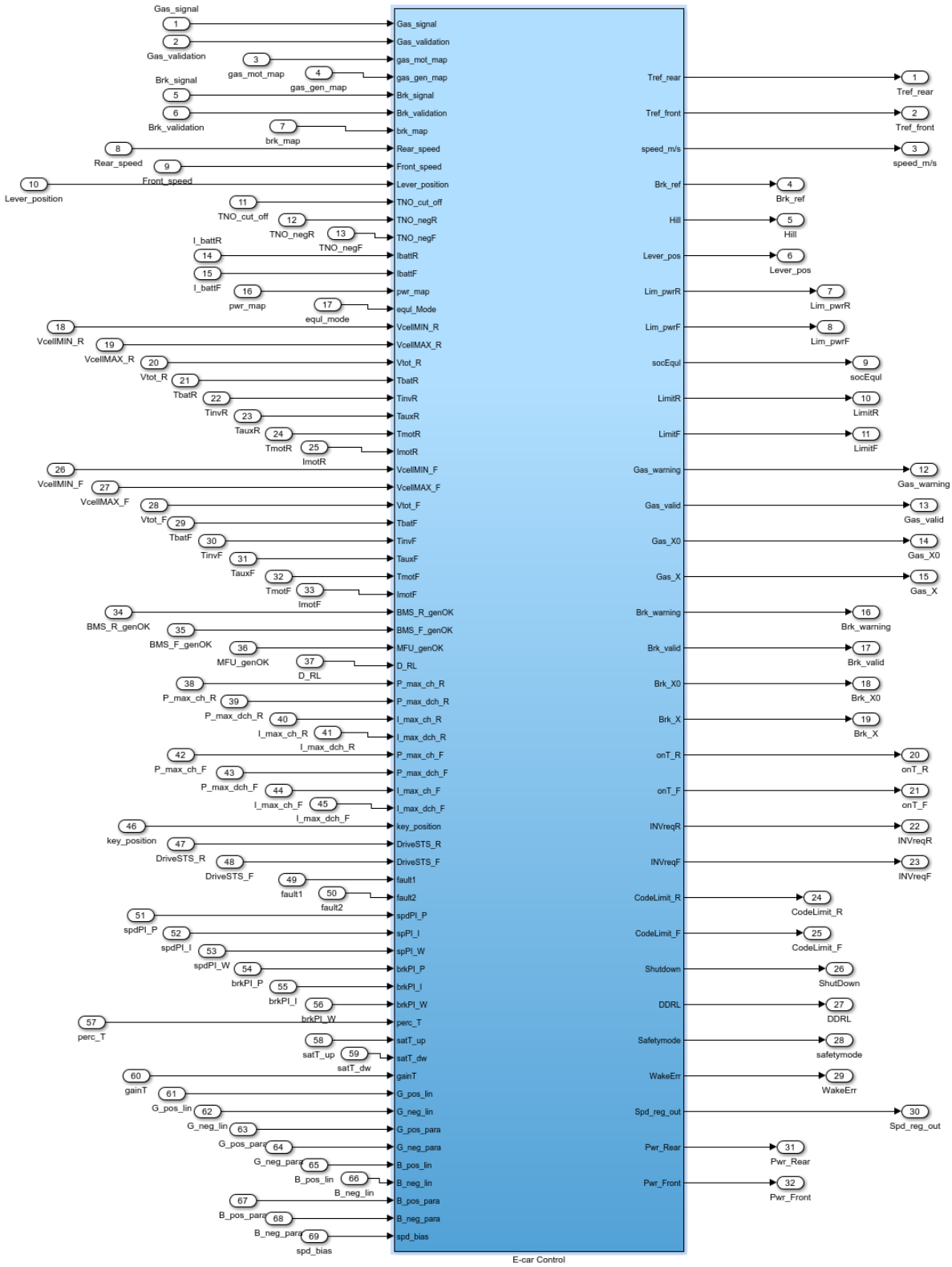


Figure 2-25 "Traction Management" algorithm block

Chapter 3
Model-based design and automatic code generation

As described in the previous chapters, a multi-drive powertrain makes available more degrees of freedom and allows to better influence the vehicle behaviour, especially regarding the drivability and the dynamics control. However, its control requires complex algorithms, which must be suitably developed and tested.

This condition is in line with the current development trend in the automotive field, where the number of electronic control units, their complexity and criticality is steadily increasing, whereas the required time and the potential error connected to manual translation of the code is no longer acceptable [14].

For several years, techniques for exploiting mathematical models for representing plants, control systems and their interactions have been investigated [15]. Additionally, the possibility to generate code to be loaded on microprocessors, starting from models used to perform system simulations [16], [17], has been a research topic as well, as part of the *Model-Based Design* (MBD) [18]–[20].

As described in Chapter 2, a *Traction Management* algorithm was developed during the Ph.D. in form of a MATLAB/Simulink model. Aiming to use the same model to both perform system simulations and program the vehicle ECU, it was decided to spend part of research in the study of the abovementioned model-based design techniques [21], [22].

The description below, is intended to provide a general discussion on the model-based design while also reporting on the AMBER-ULV case study.

Within the AMBER-ULV project, the Traction Management algorithm is integrated in the vehicle electronic control unit, along with a stability control algorithm provided by a partner of the European project, as shown in Figure 3.1. The control unit architecture will be comprehensively explained in Chapter 4.

In this chapter the outcomes of the model-based development, as carried out at the *LEMAD* laboratory, will be described. The task was performed in cooperation with *SHERPA ENGINEERING*, a French consulting company with a 15-years expertise in modelling, simulation and control design in the automotive, aerospace and energy fields.

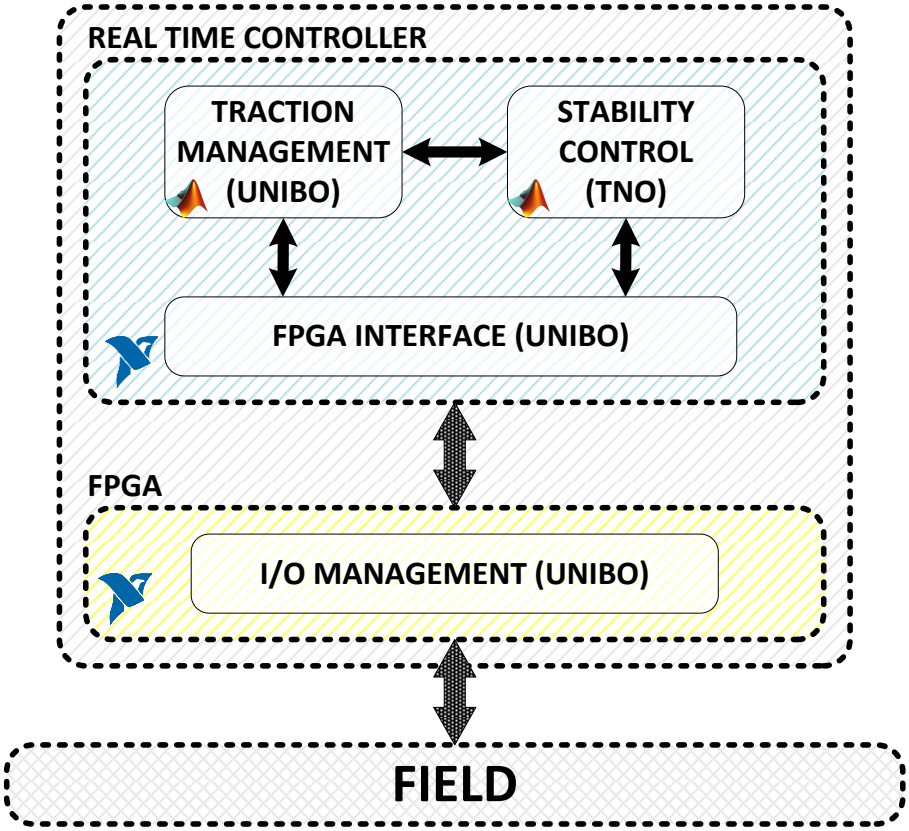


Figure 3.1 AMBER-ULV control architecture

3.1 MBD development

The model-based design is thought to provide an efficient approach for the definition of a common framework for communication throughout a design process, while supporting the *development cycle* [15].

The general MBD development cycle is usually represented through the “V-diagram”, shown in Figure 3.2. It was applied on during the Ph.D. with the necessary specifications of the project.

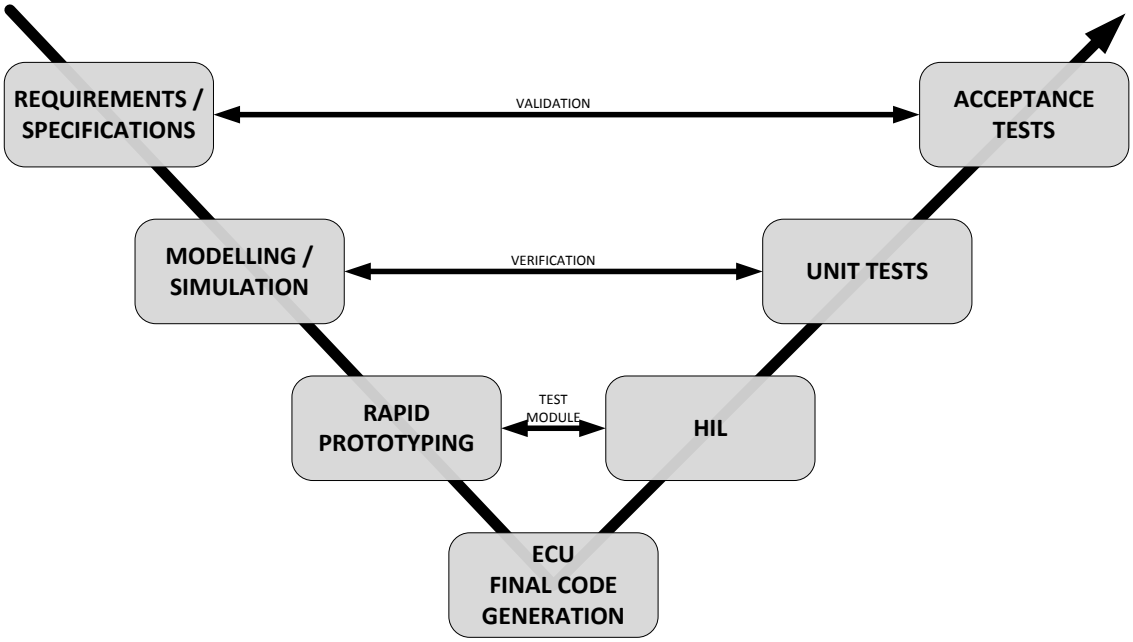


Figure 3.2 "V-diagram" of the MBD development process

The model-based design theory defines, step by step, the tasks to perform in each stage of the development process.

In relation to the control systems, the main design phases are as follows:

- **Analysis of requirements and specifications**.
- **Synthesis of a control algorithm**.
- **System and controller simulation**. For this phase, a mathematical model of the system to be controlled needs to be developed, as well as a series of “operating conditions”.
- **Controller deployment**. In this step, automatic code generation techniques significantly reduce the development time.

The abovementioned phases, which embody the first part of the diagram in Figure 3.2, lead to the completion of the system controller.

For the AMBER-ULV project, this task was entirely performed at the *LEMAD* laboratory for the Traction Management algorithm, as described in Chapter 2.

Moreover, this steps include also the *first integration*, as the Traction Management algorithm was put together with the Stability Control algorithm (provided by a partner of the European project as a compiled model) and simulated along with a vehicle model and an algorithm that simulates the driver in predefined situations, such as standardized cycles or driving profiles obtained from experimental tests. This step is represented in Figure 3.3.

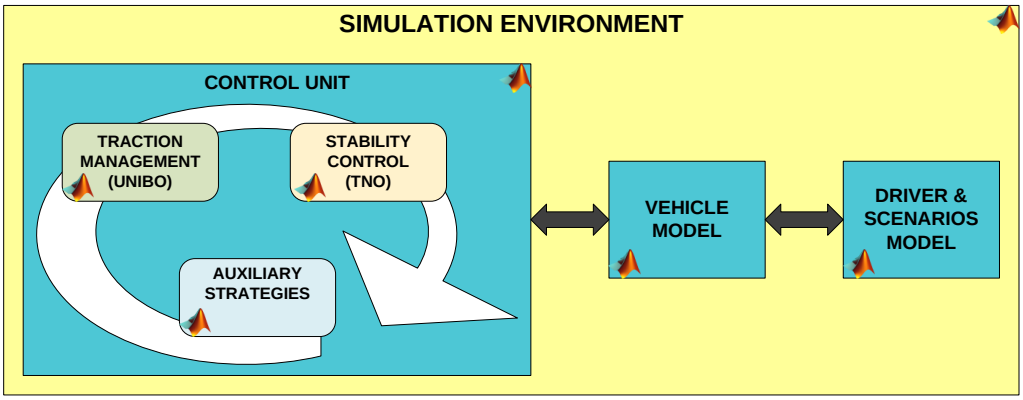


Figure 3.3 First Integration of MBD

When the control algorithm has been sufficiently tested, revised when necessary and finally validated, it can be used, through automatic code generation techniques (Paragraph 3.2), for directly programming the controller ECU. This step represents, in fact, the central block of the diagram in Figure 3.2.

According to the development cycle, the final version of the controller ECU is tested with a simulation model of the controlled system and the operating condition, yet exchanging real signals with the simulation environment. For this task, a proper interface needs to be put in place, with the purpose of simulating the real field devices to be interfaced with the controller, in terms of electrical signals, communication buses and so on. The above described phase is known as *Hardware-in-the-Loop* (HIL) test, whose scheme is presented in Figure 3.4. This analysis is very important because it allows to test the control algorithms without the real system, thus without the risk of damaging it. Moreover, abnormal situations can be safely tested as well [23], [24].

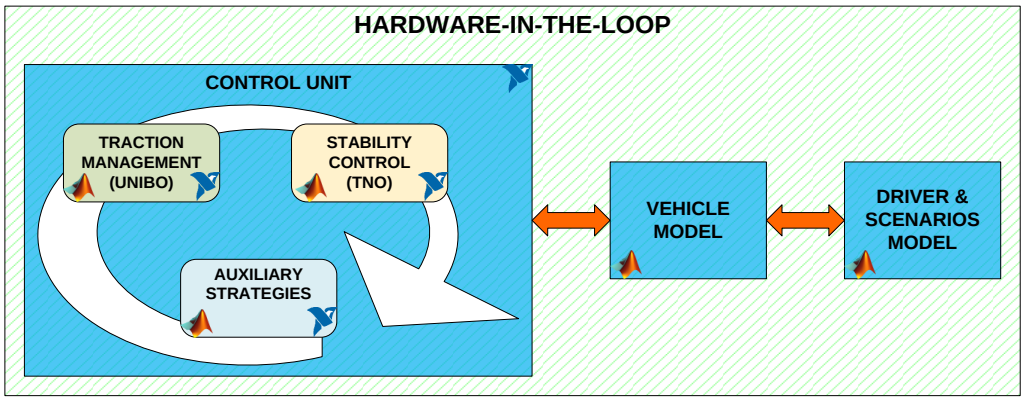


Figure 3.4 Hardware in Loop

The last step consists in the validation tests: the controller and the controlled system are tested in real operating conditions, as exemplified in Figure 3.5.

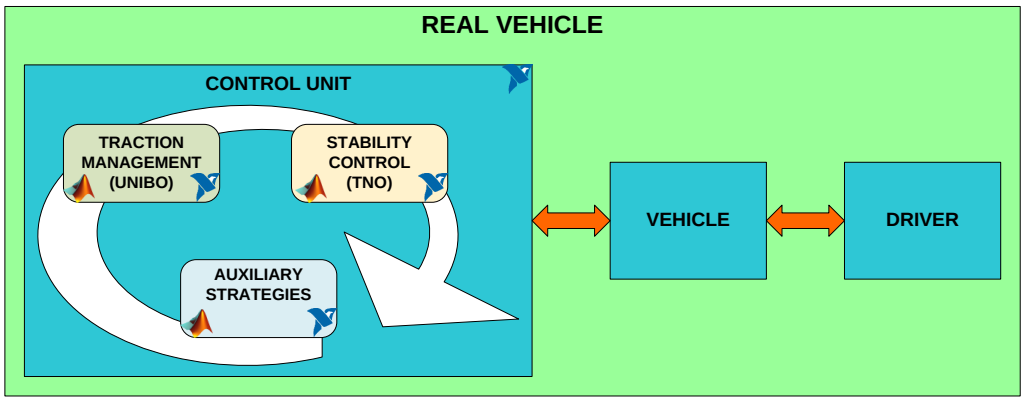


Figure 3.5 Final Implementation

Considering the AMBER-ULV project, Chapter 5 will analyse the validation tests performed on the vehicle prototype.

3.2 Development of the Simulink Model

In the following paragraphs, the process for the automatic executable code generation from the MATLAB/Simulink model is explained.

In order to develop MATLAB/Simulink models that can be suitably built for use in the LabVIEW environment, some aspects have to be taken into account.

3.2.1 Building the Simulink Model

Once the simulation model is completed, through the configuration window, the user has to make sure the *Solver* is set with a "Fixed-step" iteration mode, and a "ODE" type, as it can be seen in Figure 3.6.

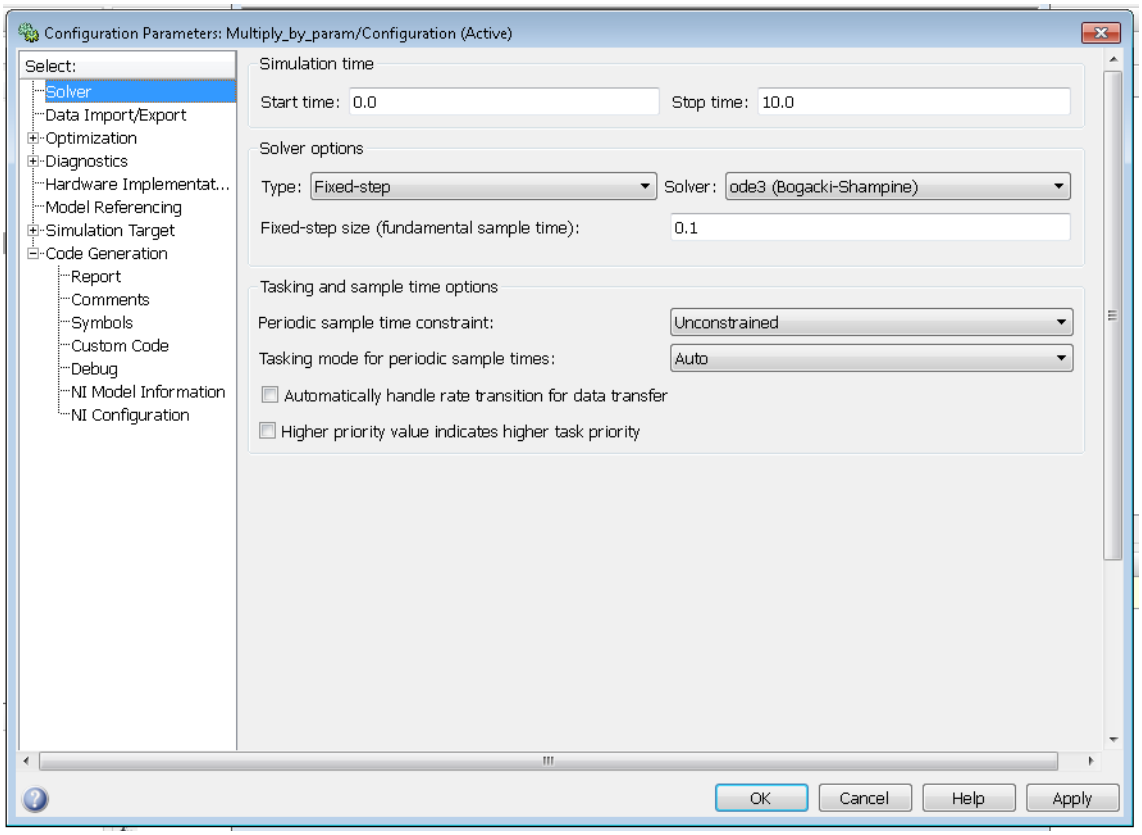


Figure 3.6: Configuration window

In the "*Code Generation*" menu (Figure 3.7), the System Target file must be selected, depending on which target (hardware) the model will be executed on.

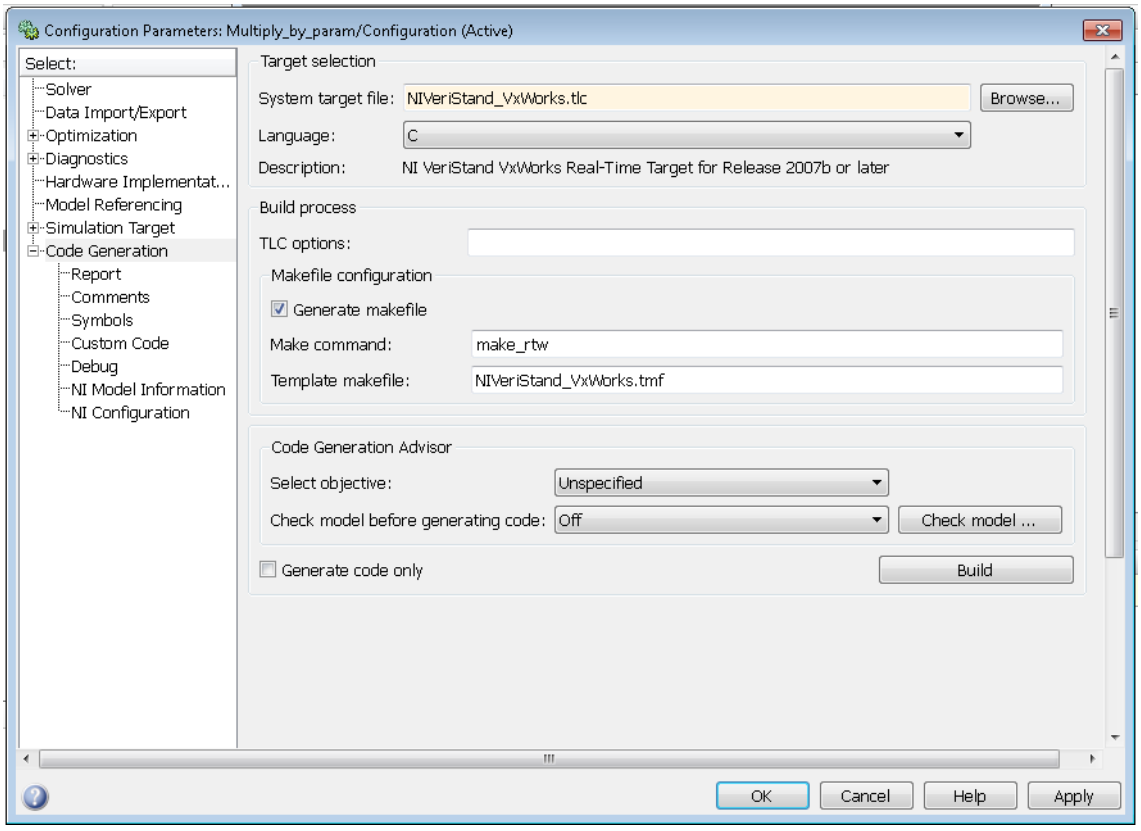


Figure 3.7: Code Generation window

By pressing the "*Browse…*" button, the *System Target File Browser* window appears as in Figure 3.8.

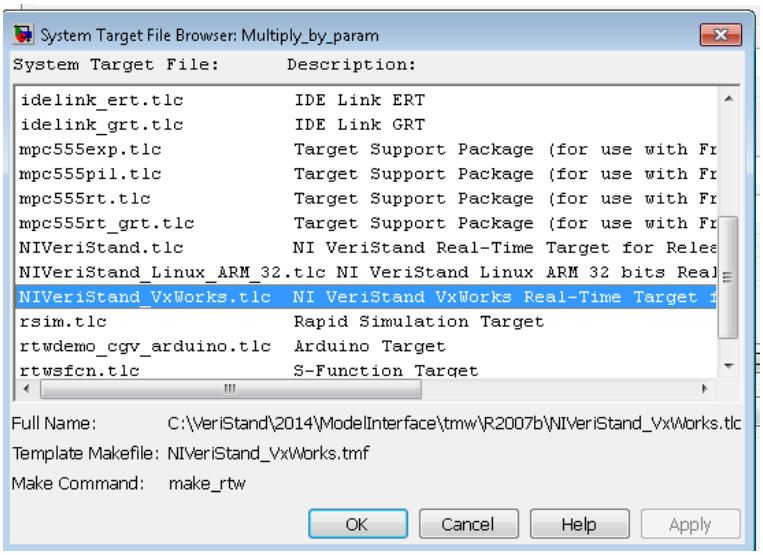


Figure 3.8: Target selection for building process

For building models for LabVIEW, the three options listed in Table 3.1 are available.

Target	System target file	Description
Windows PC	NIVeriStand.tlc	NI Veristand Real-Time Target for release 2007b or later
Linux ARM 32-bit Real-Time Controller	NIVeriStand_Linux_ARM_32.tlc	NI Veristand Linux ARM 32-bit Real-Time Target for release 2007b or later
VxWorks Real-Time Controller	NIVeriStand_VxWorks.tlc	NI Veristand VxWorks Real-Time Target for release 2007b or later

Table 3.1 Available building choices for LabVIEW target

For building the model allowing its execution on a Windows PC, "*NIVeriStand.tlc*" is the correct choice, while "*NIVeriStand_VxWorks.tlc*" builds the model in order to execute it on a Real-Time target based on VxWorks Real Time Operating System (like the CompactRIO-9022 used in the project).

3.2.2 Setting of Model's Parameters

Parameters used within the MATLAB/Simulink model can also be accessed, provided that they are properly set up in the configuration process.

As shown in Figure 3.9 in the "*Optimization*" list, the "*Signal and Parameters*" menu opens the window:

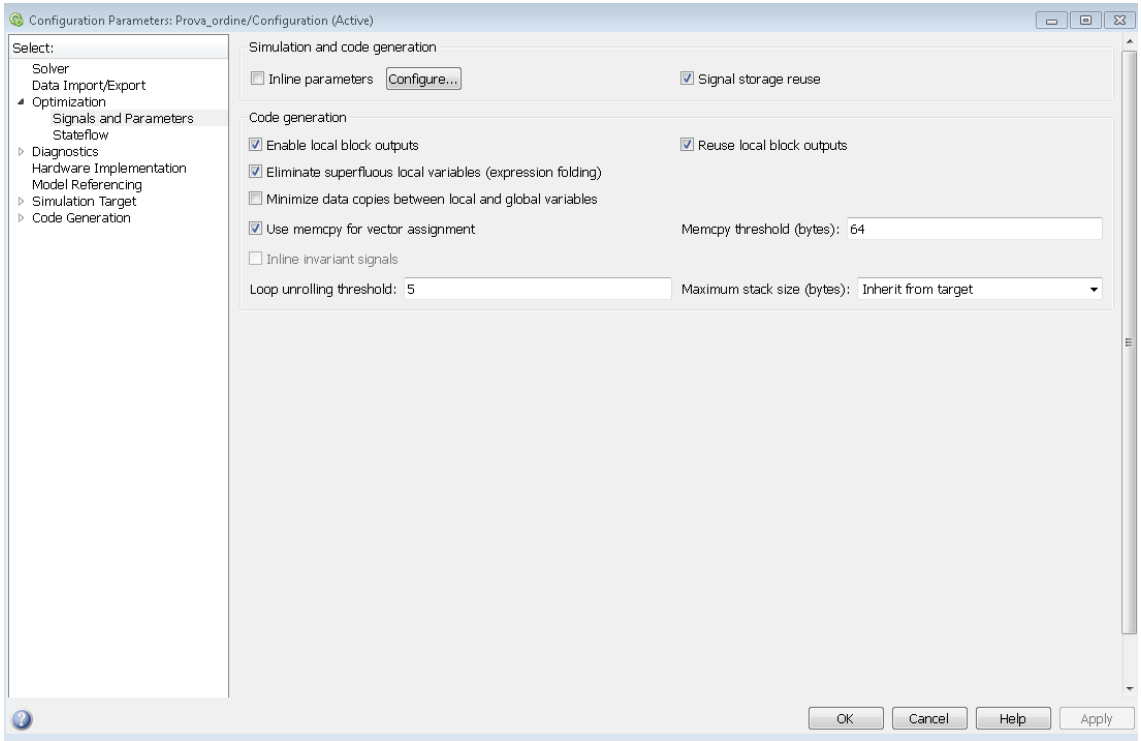


Figure 3.9 Set "inline" parameters

By clicking the "*Configure*" button (Figure 3.10), the tunable parameters setting window appears as in Figure 3.11.

Figure 3.10 "inline" parameters configuration

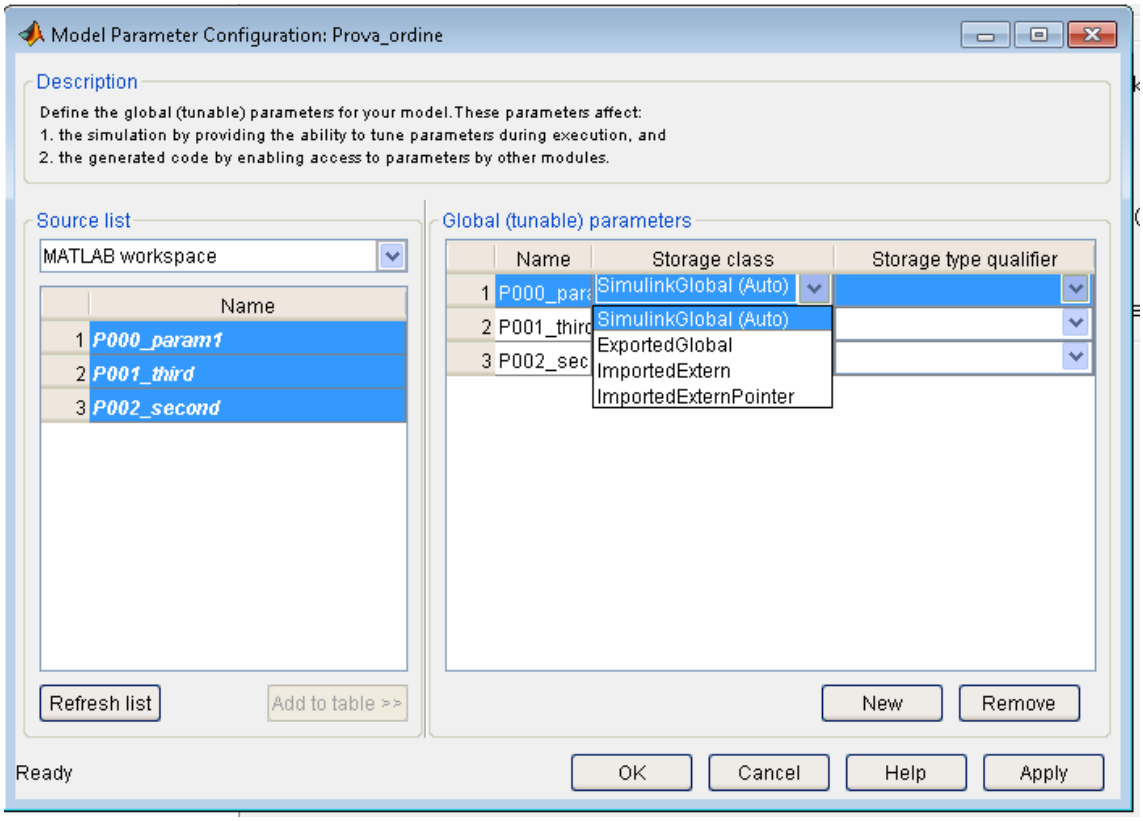


Figure 3.11 Tunable parameters setting window

The parameters present in the workspace must be added to the *Global (tunable) parameters* table.

These parameters will be accessible through LabVIEW by setting them as "*SimulinkGlobal (Auto)*" and checking "*Inline parameters*" in the window of Figure 3.10.

3.2.2.1 Parameters order

As for the input and output ports (see Paragraph 3.3), the LabVIEW *Model Interface Toolkit* treats parameters as a double precision floating point array, whose order reflects the order of the MATLAB workspace, which is alphabetical.

Simple models have been developed in order to test how the compiler queues multiple parameters and how it behaves when a model is updated adding or removing parameters (as in Figure 3.12).

In order to prevent incorrect address of parameters (which may not be programmatically detected) parameters are defined with an alphanumerical code in the form "*P0000_param-name*", allowing the developer to keep control of the alphabetical order by properly setting the value of the four digits (parameter number), as well as maintaining the readability of the model by including a brief description of the parameter function in its name.

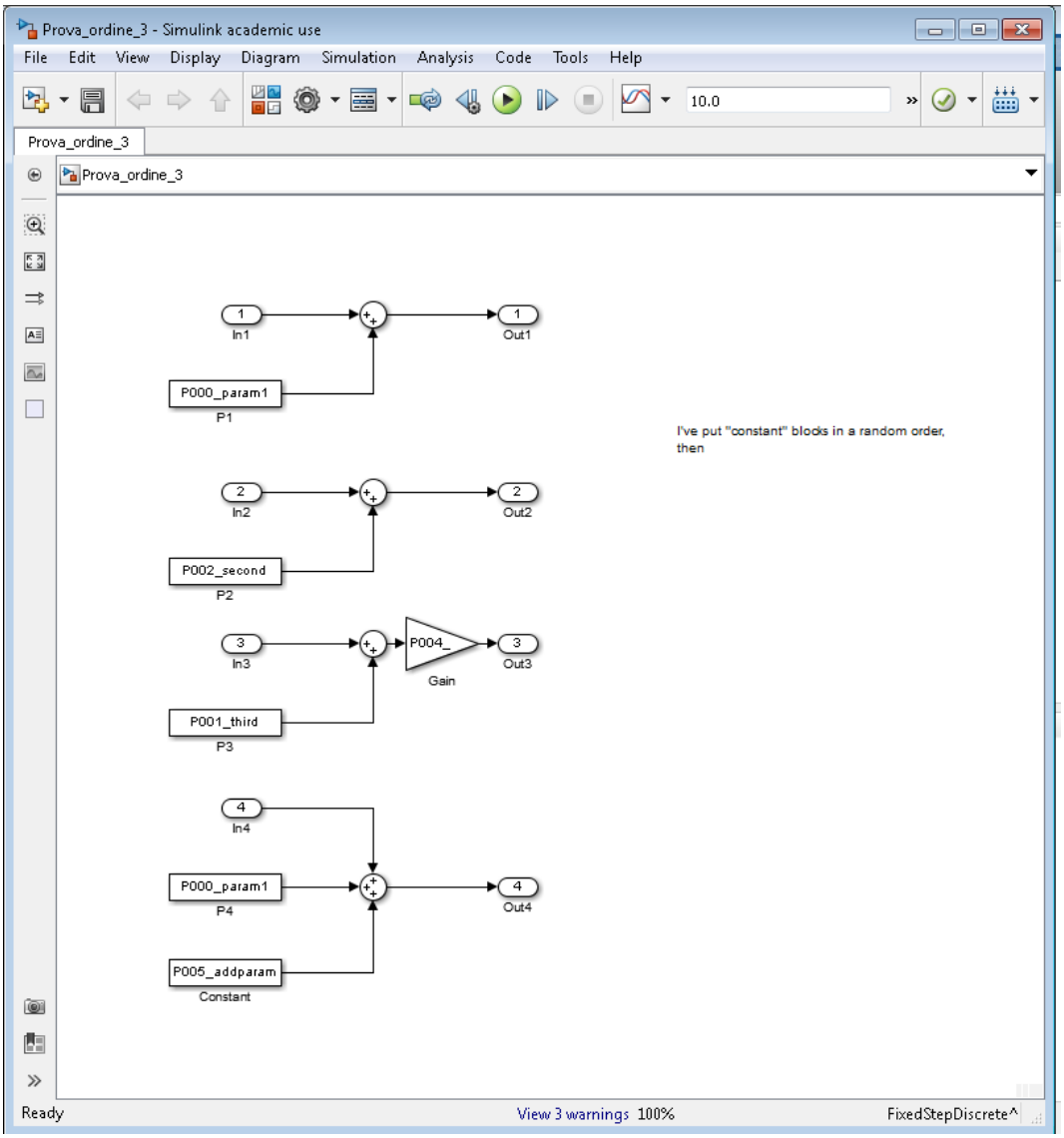


Figure 3.12 Test example for parameter addressing

3.3 Using the Simulink Compiled Model in LabVIEW

3.3.1 Target preparation

The model compiled within the *Simulink* environment must be in the target memory for the interface functions to work. For Windows PC targets, the model file simply needs to be in the same PC memory, and the path control will point to this file.

For Real-Time targets, the compiled file must be copied into the execution target's memory, by accessing it through an FTP protocol as shown in Figure 3.13:

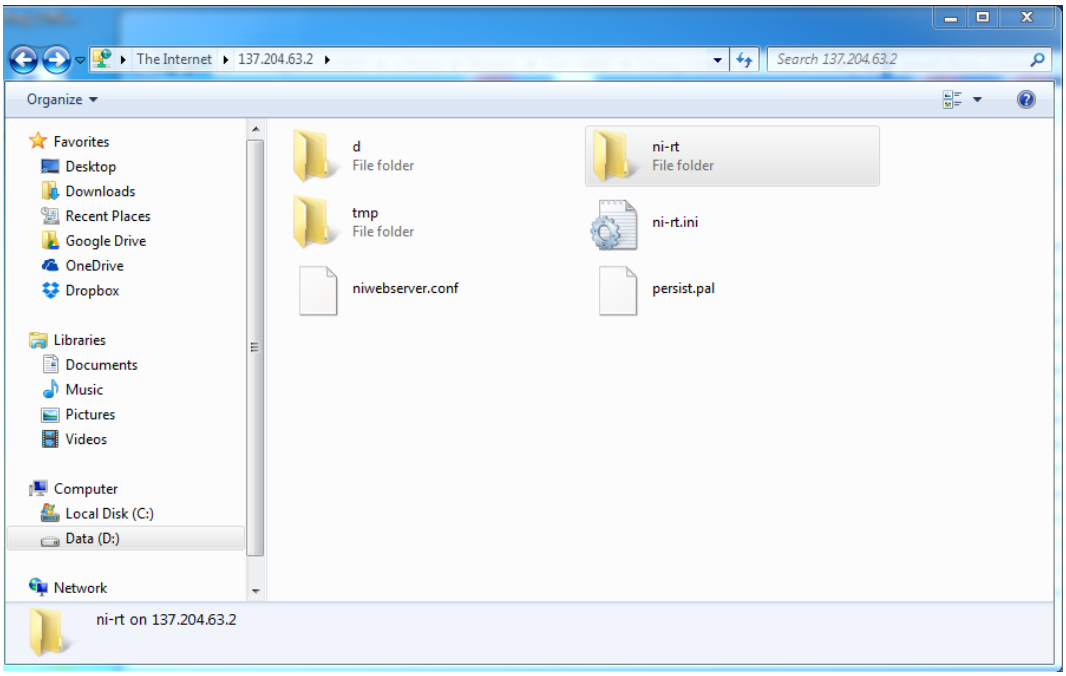


Figure 3.13 Access to the target memory through FTP

The correct path for the *CompactRIO* execution example is "*FTP -> IP.address -> ni-rt -> system -> filename.out*".

The *Model Interface Toolkit* must also be installed on the execution target (Figure 3.14).

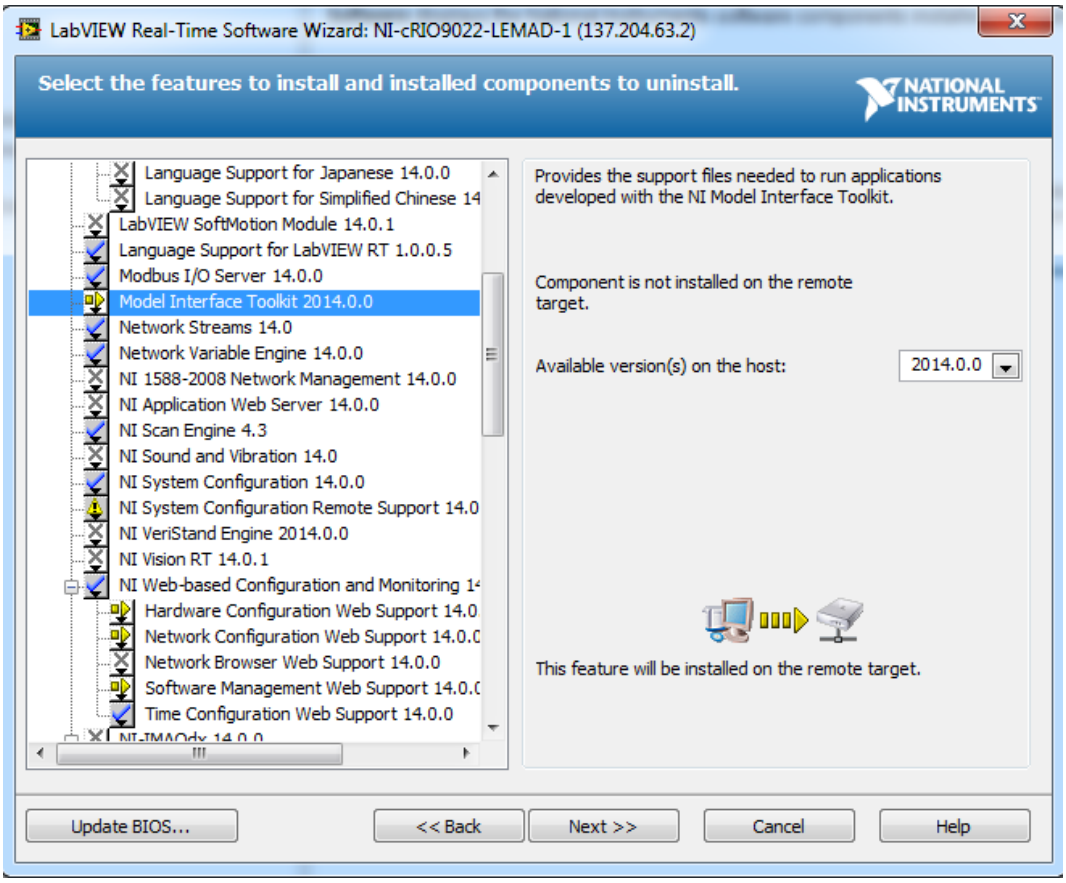


Figure 3.14 Model Interface Toolkit installation on a Real-Time target

3.3.2 *Executing compiled* models with Model Interface Toolkit

The models compiled as described in (Paragraph 3.2) can be executed in the LabVIEW environment thanks to the *Model Interface Toolkit* (MIT).

Figure 3.15 shows a typical programming structure for integrating a compiled model within the LabVIEW environment using *MIT*.

As it can be seen, there are a few “single shot” functions at the beginning and at the end of the code execution (described in Table 3.2) and two loops, which are described further on.

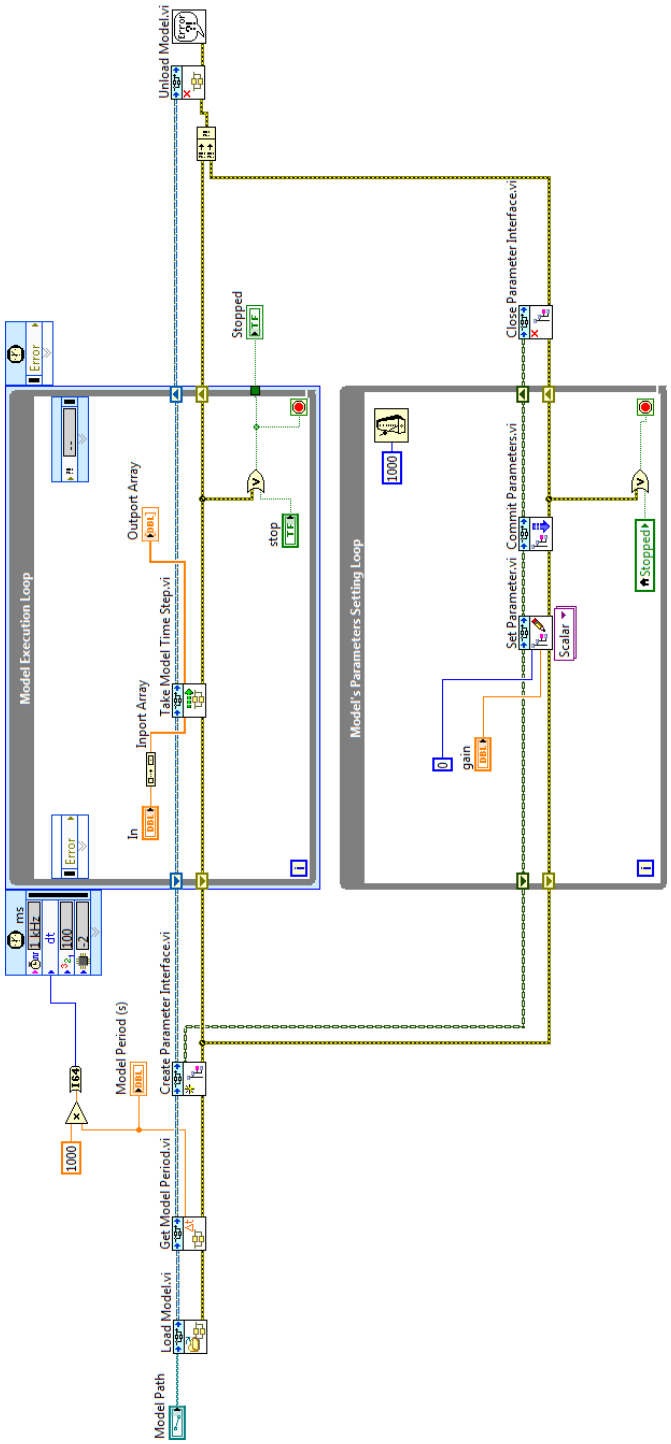


Figure 3.15: Model Interface Toolkit typical programming structure

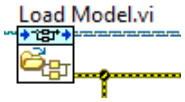
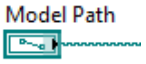
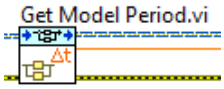
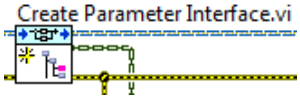
Function block	Description
	Loads a model into memory and prepares it for execution. This Vi must be called to load a model before you can execute it in a control loop.
	The path to the model to be prepared for execution is specified with this control. Models must conform to the NI VeriStand Model Framework header file (NIVERISTAND_API.h). Otherwise, LabVIEW returns error code "-383200".
	Returns the rate, in seconds, at which the model was compiled to run. A model is set to run at a certain rate, or step size, as defined in the build options when the model is compiled. However, the rate at which the Model Interface API actually steps the model is determined by how often the *Take Model Time Step VI* executes in the application.
	Creates a reference for the parameters manipulation functions, to initialize or update them from background-priority code as the model executes.
	Closes all the references with the model and discards it from the target memory.

Table 3.2 Model Initialization and finalization functions of MIT

The code snippet of Figure 3.16 contains the *Take Model Time Step* function, which actually runs one iteration of the compiled model each time it is called. For the correct behaviour of the compiled model, the loop iteration period is set at the same time step specified in the model properties.

Both the input and the output of this function are floating point double precision arrays, the order of which corresponds to the "port number" set within the Simulink model. With this representation, there is no information about the port name. Therefore, an appropriate strategy must be put in place to avoid errors in the assignment of inputs and outputs (see chapt 4). This has been achieved with careful planning of the variables used and available during every integration step encountered.

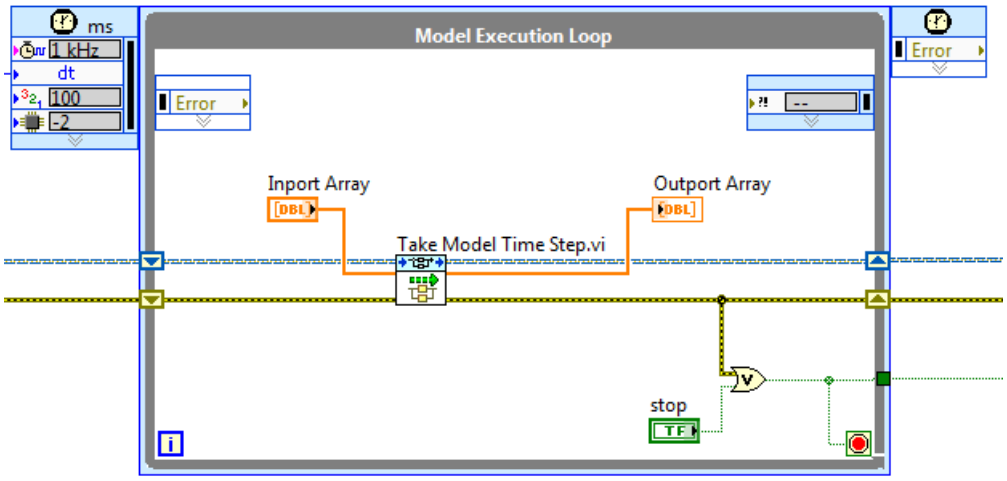


Figure 3.16 Model Time Step of MIT

Additionally, model parameters can be read and written at run time through the function displayed in Figure 3.17.

This function runs in background, thus allowing the use of a standard loop with a low iteration frequency, as the tuning of parameters is not supposed to require high determinism and speed.

As well as the inputs and outputs, parameters are treated as “DBL” array, the order of which corresponds to the alphabetical order of the parameters in the MATLAB workspace. Thus, a similar strategy must be set up to prevent assignment mistakes.

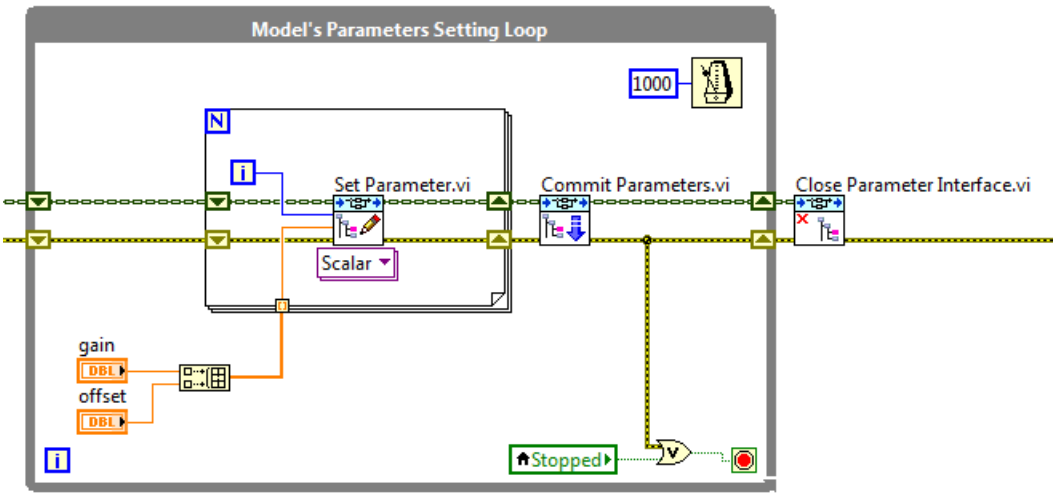


Figure 3.17 Access to model parameters through MIT

Chapter 4
AMBER-ULV System Integration

4.1 Introduction

The research activity conducted in the field of *multi-drive* powertrains has also been applied to accomplish the goals of the European project "AMBER-ULV". The target architecture of the project called for a single vehicle control unit, developed on a rapid prototyping hardware, named *Unified Control Platform* "UCP".

To achieve this purpose, the traction management functions reported in Chapter Chapter 2, together with the code generation techniques described in Chapter Chapter 3, were exploited to implement the high-level algorithm of the vehicle control unit.

This chapter deals with another fundamental task of the AMBER-ULV project, that is the *system integration*. This statement refers to all the development activities necessary to ensure that all the hardware and software components can function together as a system. For the AMBER-ULV project, it includes all the design tasks related to the hardware connections between the "field" and the ECU, and the low-level software which exchanges, with high-level algorithms, the required information for the system to operate as expected.

A *National Instrument Compact RIO* was chosen as the rapid prototyping platform. It is a modular platform consisting in a Real-Time microprocessor and a *Field Programmable Gate Array* chip (FPGA) that can be connected to different *modules* depending on the type of signal to be acquired or generated. The actual device used is shown in Figure 4.7.

The many sensors necessary to monitor and effectively control the vehicle behaviour needed to be carefully selected and their signals conditioned to satisfy the UCP acquisition modules and the range of operation desired.

In general, there are three types of signals produced by the sensors that are connected to the UCP through specific acquisition modules: analog, digital and serial *CAN-Bus*. As it will be described later, some sensors can also produce more than one signal type. For example, the gas and brake pedals have both digital and analog outputs.

After the hardware conditioning, some software manipulation on the acquired signals is still required to properly prepare them for the control algorithms.

In the following, the implemented development process will be described, itemizing those aspects that have affected the design choices.

In particular, the following tasks will be analysed:

- Analysis of necessary information from the field. This include the selection of the sensors, the development of a test bench for preliminary testing them with their acquisition chains and the hardware design for signal conditioning;
- Real Time Controller;
- Definition of signals and their processing in UCP;
- Implementation of Low Level Software in UCP;
- Acquisition and Communication tests on AMBER T3 demonstrator.

4.2 General Considerations and Sensor Selection

4.2.1 Analysis of necessary information from the field

According to the vehicle features described on the Project Description Of Work, several information from the field are necessary for the correct UCP operation.

The Battery Management Systems (BMS), the Traction Inverters and the electronic differential controller (TCSS), were designed providing CAN-Bus connection with UCP. It means that said devices are in charge of the acquisition, the signal conditioning, the scaling and any other needed signal processing. All information is then exchanged through the serial bus and is "ready" to be used within the UCP.

The following signals, on the contrary, are not transmitted through any communication bus and must therefore be acquired directly by the UCP:

- Gas and Brake Pedal Position;
- Direction ("Gear") Lever Position;
- Traction Motor Temperature(s);
- Electronic Differential Controller "TCSS" Temperature(s);
- Brake Circuit(s) pressure;
- Wheel(s) Speed.

Moreover, the following signals are required for the proper operation of the stability control algorithm:

- Three Axis Accelerometer;
- Three Axis Gyroscope;
- Steering Angle.

Considering the Steering Angle, the Electric Power Steering used on the prototype is equipped with a CAN interface through which the required information (Steering Angle, status and diagnostic signals) is broadcasted and commands are also received.

The Acceleration and Gyroscope signals, as it will be described later on, come from a device equipped with a UART serial communication. A dedicated level adaptation and CAN interface circuit was developed for this purpose.

4.2.2 Selection of Sensors for the AMBER prototype

In this section, the sensors used on the AMBER vehicle are listed.

4.2.2.1 Gas and brake pedal position

Both the Gas and Brake Pedal Sensors are based on *Start STL1D* Hall effect rotational angle sensor. This sensor has a voltage output proportional to the angle, within the range $0.5 \div 4.5\ V$. Additionally, it provides, as most commercial pedal sensors do, the "IVS" digital signal (Idle Validation Signal), which is used as a safety feature: the IVS signal is 0 when the pedal is

completely released and up to a given threshold, usually 10% of the full pedal travel, and it turns into 1 if the stroke exceeds the aforementioned threshold.

4.2.2.2 Direction ("Gear") lever position

The *Direction lever* is a command intended to be used by the driver as a driving direction selector (Forward – Reverse), as well as a driving mode selector (i.e. "Eco" – "Sport") as it provides two different "Drive" positions. This command is based on the *Start SCE07* hall effect rotational angle transducer. The lever position can be determined as combination of four digital signals.

4.2.2.3 Traction Motor Temperature(s)

The motor temperature is measured by means of a NTC thermistor enclosed in the stator windings, connected as a leg of a voltage divider. The component used is the *Amphenol Advanced Sensors DKF103N3*.

4.2.2.4 Stability Control Actuator "TCSS" Temperature(s)

The TCSS friction discs are mounted on the same frame as the final drive reducer, and share the lubricant oil with the gear set. Therefore, the gearbox oil temperature is measured through an NTC thermistor as a condition monitor variable. The selected sensor, connected as the lower leg of a voltage divider, is the *VDO 323-801-017-001K/N*.

4.2.2.5 Brake Circuit(s) pressure

The AMBER vehicle braking system is designed with two independent hydraulic circuits with independent pumps actuated by a common pedal. Each braking circuit is also equipped with a piezo-resistive pressure transducer *AVIORACE SP100M10x1*.

4.2.2.6 Wheel(s) Speed

The speed of each wheel is measured on the AMBER-ULV prototype as part of the necessary information for the vehicle state estimation which is, in turn, an essential information for the stability controller to be effective. The wheel speed is measured thanks to a phonic wheel fitted to the wheel hub and a *OMRON E2A-S08KS02-WP-C1* proximity sensor. The algorithm implemented for the wheel speed estimation is described in paragraph 4.6.2.5.

4.2.2.7 Three Axis Accelerometer and Gyroscope

The accelerations and rotation rates of the vehicle are also part of the required information for the stability control algorithm. These measurements are obtained thanks to an Inertial Measurement Unit (IMU) that broadcasts, over a serial RS232 bus (UART standard), the accelerations over three perpendicular axes (x, y, z) and the rotation speeds with respect to those axes.

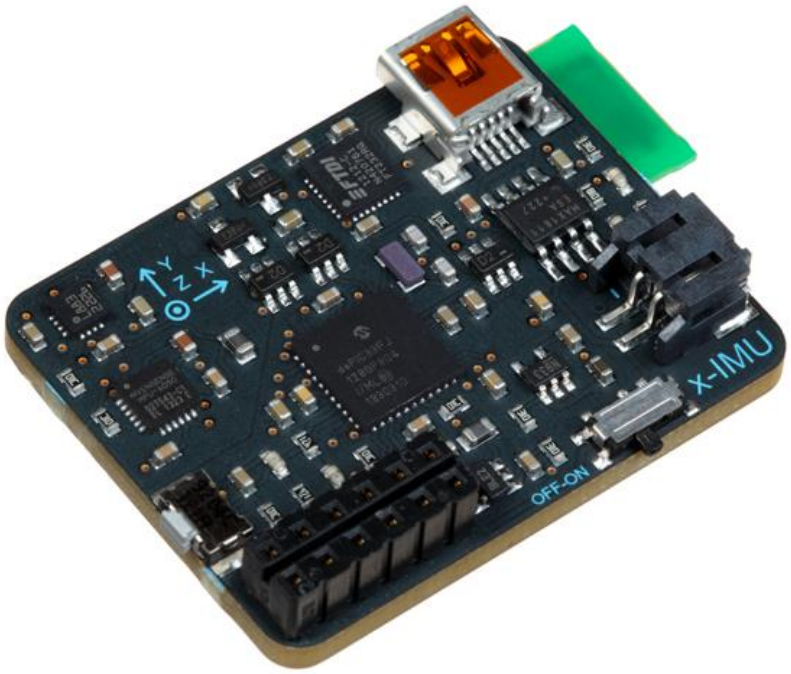


Figure 4.1 Inertial Measurement Unit employed on the AMBER vehicle

Its noteworthy features, as regards of the AMBER project, are here summarized:

- Triple axis 16-bit gyroscope - Selectable range up to ±2000 °/s;
- Triple axis 12-bit accelerometer - Selectable range up to ±8 g;
- Factory calibrated;
- Temperature compensated (gyroscope only);
- Selectable data rates up to 512 Hz.

4.3 Sensors Preliminary Test Bench

All the above listed sensors were tested at the *LEMAD* laboratories before the final assembly on the AMBER-ULV vehicle, with the goal of verifying the real sensor compliance with the provided specifications and experimentally tuning the prototype signal conditioning circuitry.

Significant examples are reported below.

4.3.1 Direction lever sensor

Figure 4.2 shows the preliminary test setup realized for the direction lever.

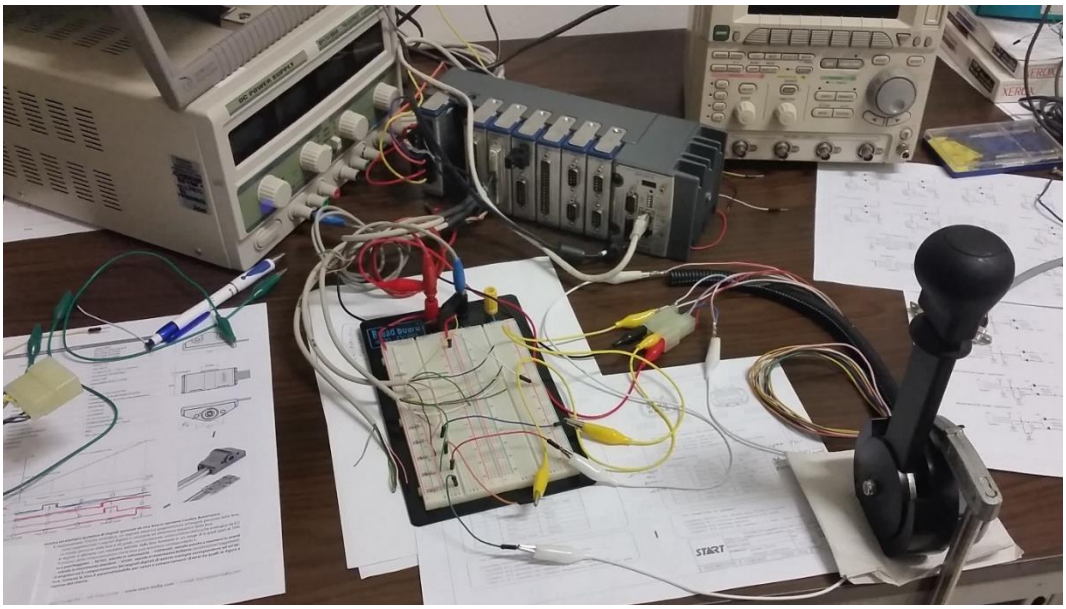


Figure 4.2 Direction ("Gear") Lever Position sensor preliminary test

This angular sensor presented some differences compared to the datasheet specifications. In particular, the digital "PARK" signal wire was not installed, so that four digital signals were brought to the UCP instead of five. The "FORWARD" signal has been found to differ from the datasheet specifics, but it was of no consequence for the overall functionality of the lever.

Test results and differences are summarized in Figure 4.3. The green lines represent the datasheet specifications, that were probably referred to a similar sensor of the same manufacturer. Red lines are the preliminary test results obtained at LEMAD.

The parking brake active condition is detected thanks to a micro switch fitted on the parking brake lever. This signal, however, was not managed by the UCP in its first release.

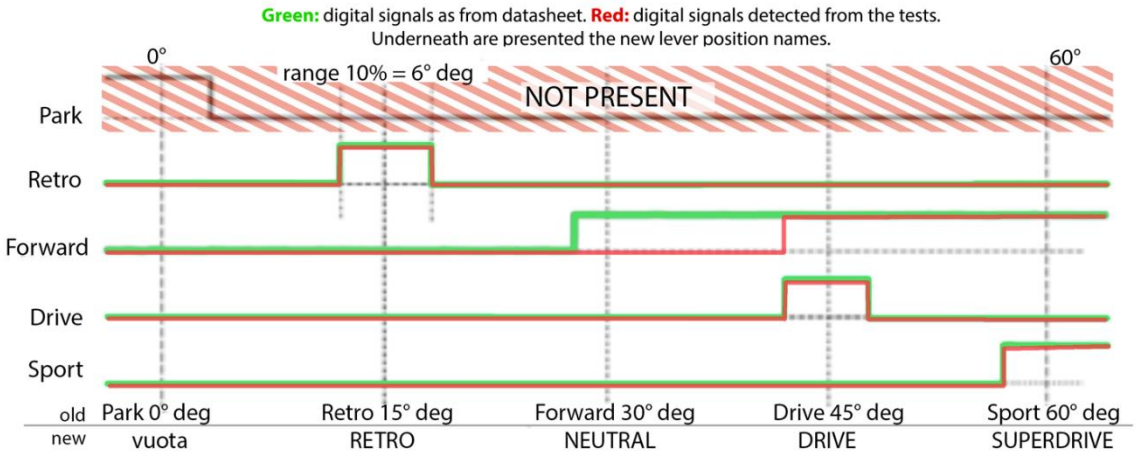


Figure 4.3 Direction Lever's demonstrator tests with differences from the datasheet outputs

4.3.2 Wheel speed measurement system

Figure 4.4 reports tests made by installing the wheel speed sensor and its phonic wheel on a lathe, in order to check the performance of such a measurement system and to prove the correctness of the conversion functions.

The wheel speed estimation algorithm is described in 4.6.2.5.

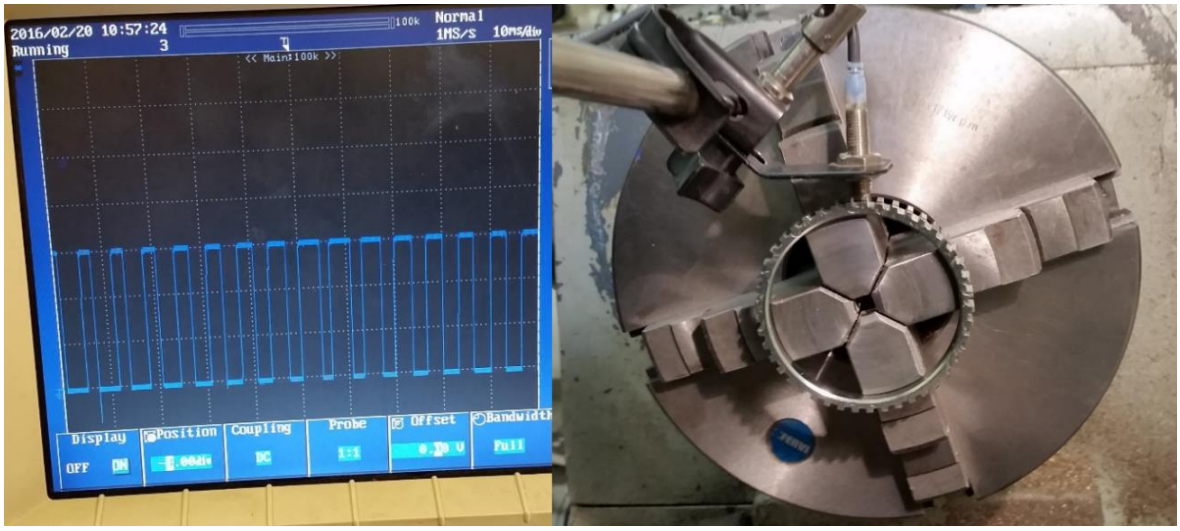


Figure 4.4 Wheel Speed sensor preliminary test

4.3.3 Inertial Measurement Unit UART to CAN interface

Considering the acquisition of the quantities coming from the IMU device, the first investigated approach consisted in the connection of the *x-IMU* serial bus to the UCP real time controller RS232 port. For this purpose, a level conversion circuit was developed (see left side of Figure 4.5) in order to adapt the *x-IMU* output (3.3 V UART standard) to the *cRIO* serial port (RS-232 standard). Figure 4.6 shows the input and output signals of the developed circuit: the yellow signal represents the *x-IMU* 3.3 V UART output, while the pink one is the ±5 V RS-232 output.

The firsts tests conducted on the described configuration proved that this execution is not as effective as expected, because of the lack of available documentation and libraries for the implementation of the needed functions on a NI real-time target [25].

In order to achieve a better result, after literature analysis, the development of an external RS-232 to CAN interface seemed to be the right choice.

As a first trial, such a device was simulated on a Windows computer, using the NI LabVIEW programming language, a RS-232 to USB and a NI CAN-USB interface.

Figure 4.5 shows the test of the Inertial Measurement Unit, the UART to RS-232 converter developed at LEMAD and the software simulated Serial to CAN interface developed in LabVIEW.

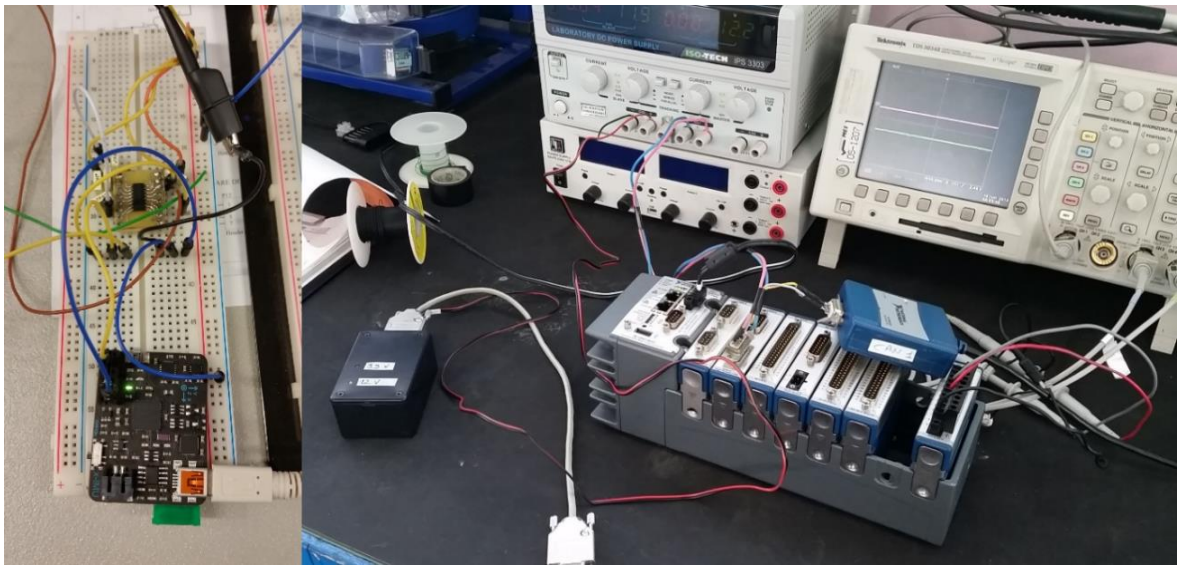


Figure 4.5 Three Axis Accelerometer and Gyroscope preliminary test

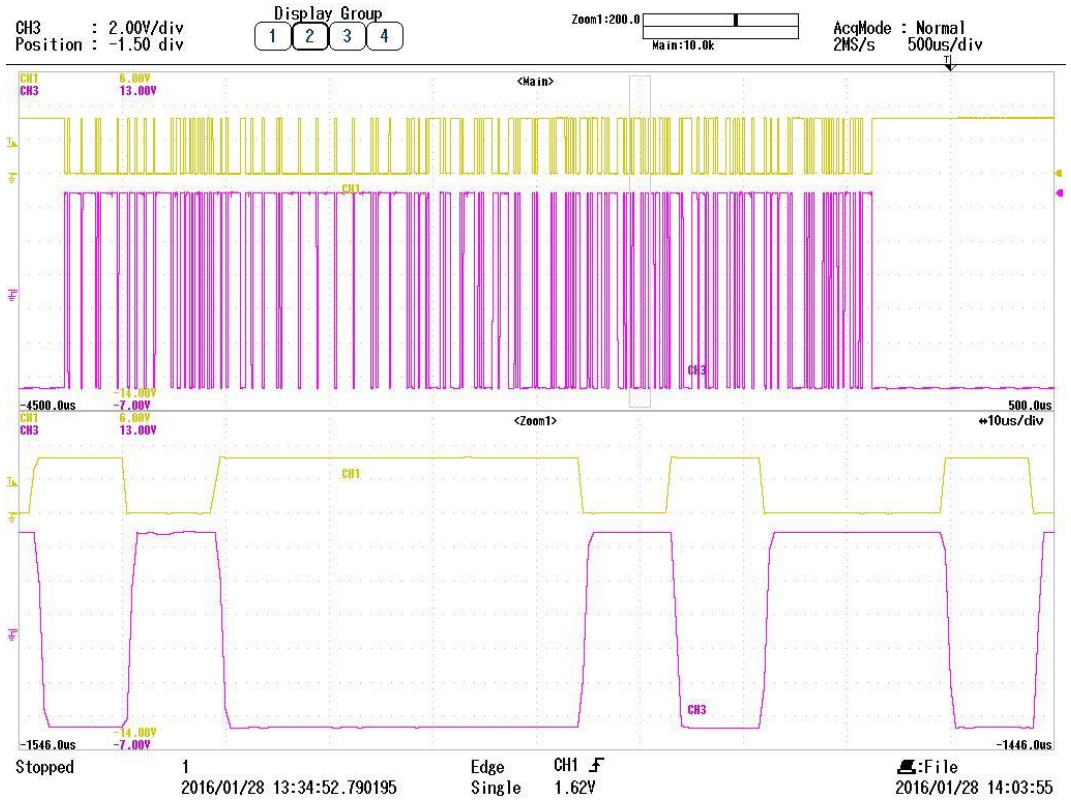


Figure 4.6 UART to RS-232 converter, I-O comparison

4.4 UCP Platform

4.4.1 Real Time Controller

As anticipated, the chosen solution for the real-time controller was the National Instruments *Compact-RIO*: a rapid prototyping platform that combines a VxWorks Real-Time Microprocessor with an FPGA, obtaining a high determinism of the developed code, as well as the possibility to customize the I/Os by means of specific I/O modules, according to the needs.

The actual platform is the "NI cRIO-9022", that comes with a 533 MHz Real Time processor, 2 GB of non-volatile memory and a 256 MB DDR2 memory.

4.4.2 I/Os

The assessments carried out showed that the UCP must be able to interface with the following signal types:

- CAN Bus (4);
- Analog Inputs (8);
- Digital Input (6);
- High Speed Digital Input (4);
- Digital Output (1);
- Analog Output (4);

The numbers in brackets represent the required number of channels for each signal type.

Up to four Analog Outputs were also added and used during the debug processes, although not required by the UCP tasks related to the traction management.

According to what previously stated, the following National Instruments "C-Series" I/O modules were chosen to be mounted on the cRIO chassis:

- 2x NI 9853 – 2 Ports High Speed CAN;
  a two ports high speed CAN module, based on the Philips TJA1041 transceiver, for which the maximum baud rate is 1 Mbps.
- 1x NI 9205 – Analog Input;
  used in the range ±10 V, it's a 32 channel (single ended) 16-Bit Analog Input Module.
- 1x NI 9411 – High Speed Digital Input;
  this is a 6-Channel Differential Digital Input Module (even though used in single ended configuration for the AMBER UCP), with high sample rate, in the order of magnitude of microseconds.
- 1x NI 9425 – Digital Input;
  provides 32 sinking digital inputs, 24 V maximum.
- 1x NI 9476 – Digital Output;
  32-Channel, 24 V, sourcing digital output module
- 1x NI 9263 – Analog Output;
  this module, used to check the correct operation of the control algorithms in the debug phase, has four Analog Outputs, ±10 V, 16 Bit, 100 kS/s/ch simultaneous.

Figure 4.7 shows the final configuration of the AMBER control platform, also known as the Unified Control Platform.

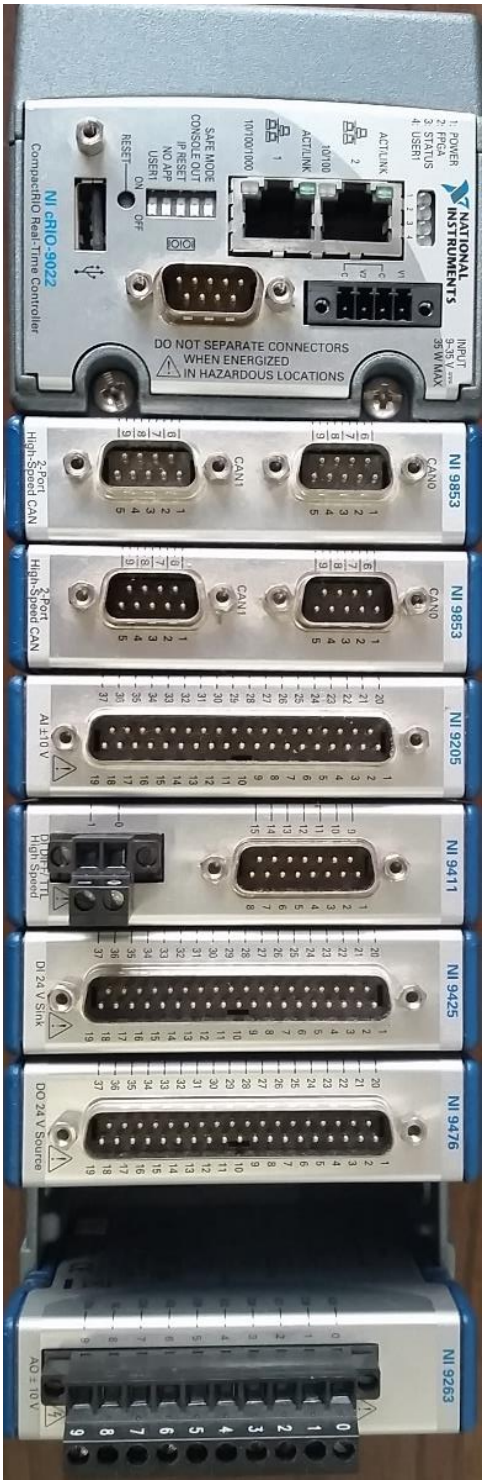


Figure 4.7 AMBER UCP based on NI cRIO Platform

4.4.3 Hardware

Considering the characteristics of the sensors used for the prototype and those of the UCP I/O modules, it was necessary to develop a dedicated hardware for interfacing the sensors with the UCP and for obtaining the appropriate signal conditioning for each type of sensor.

In this chapter, the hardware interface, as designed by the writer, is presented with reference to the main UCP connections with sensors.

A 12 V power supply is provided to the UCP box (Figure 5.2). Also in this box, a voltage regulation circuit provides a 5 V power supply for the sensors requiring it.

4.4.4 Analog signals

4.4.4.1 Gas and Brake analog signal

Figure 4.8 shows the connections with the Gas and Brake pedal position sensors. As described in 4.2.2.1, the pedal sensor provides both an analog and a digital signal. This means that they share the same connector.

The digital signal and the analog signal of the gas pedal are marked, in Figure 4.8, with “DI1” and “AI1” label, respectively. The same goes for the brake pedal, with “DI2” and “AI2”.

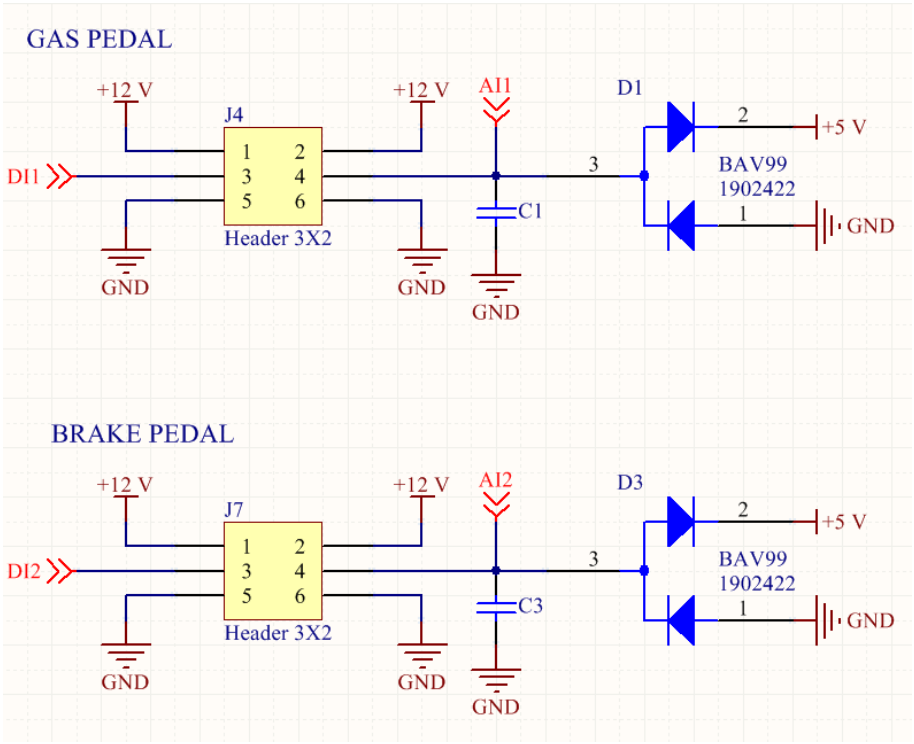


Figure 4.8 Gas and Brake pedal connection

“C1” and “C3” represent a hardware arrangement for mounting a filter capacitor in case of excessive noise on the signal. On the AMBER-ULV prototype no abnormal disturbances were detected, therefore the filter capacitors have not been mounted.

The double diode “BAV99” protects the analog input of the acquisition module from signal values outside of the range 0-5 V.

4.4.4.2 Motor temperature signals

The temperature sensor is a resistor whose value varies with the temperature. The thermistor is connected as a leg of a voltage divider and the intermediate voltage is temperature-dependent according to the thermistor characteristic and the voltage divider equation.

Figure 4.9 reports the interface circuitry for the motor temperature measurement. The connection is similar to the one presented for the pedals, except for resistors “R2” and “R4”, which are the fixed legs of the voltage dividers.

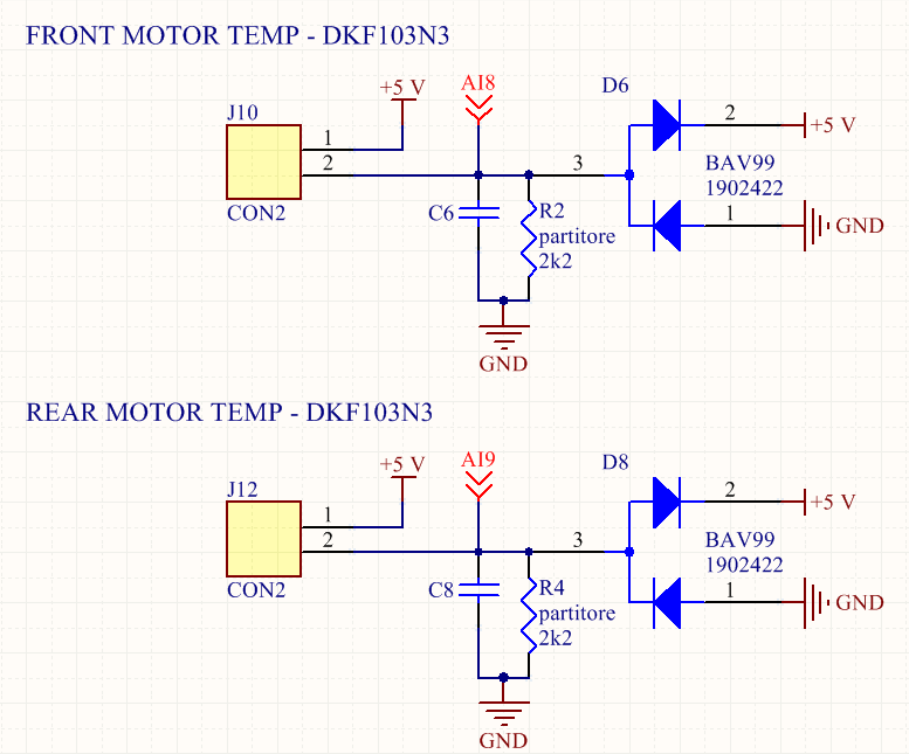


Figure 4.9 Front and Rear motor temperature sensor connection

The circuit designed for TCSS temperature measurement is basically identical to the one here described, so it is not reported hereinafter.

4.4.5 Digital signals

4.4.5.1 Gas and Brake validation signals

As described for the analog signals, the digital signals mounting arrangements for filter capacitors and signal range limiters are also provided.

In this case, a pull-up resistor is required and it can be seen labelled as “R12” and “R13” respectively in Figure 4.10.

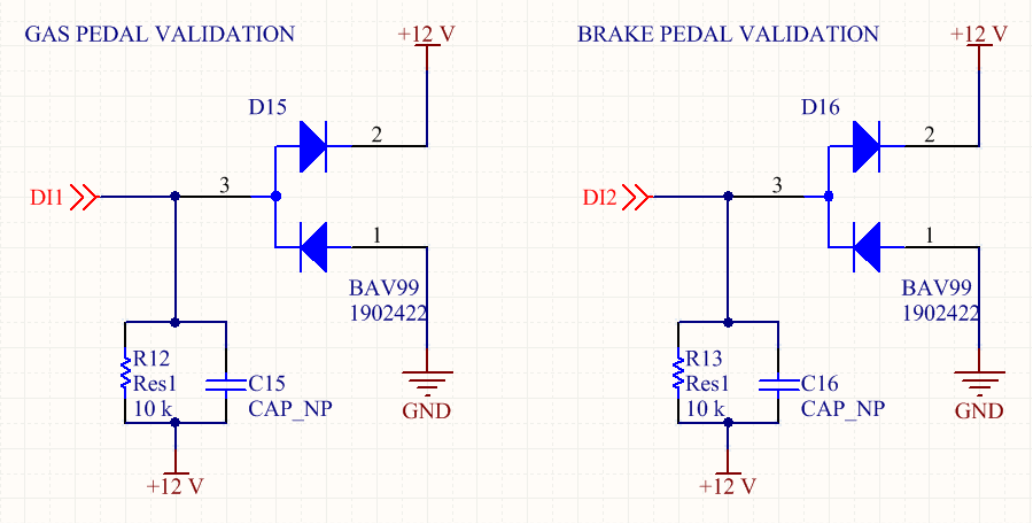


Figure 4.10 Gas and Brake pedal digital signals

4.4.5.2 Direction lever signals

As described in 4.2.2.2, the lever position sensor provides four digital signals.

Their connections with the UCP, shown in Figure 4.11, are identical to those described for pedals validation signals. The signals required a pull-up connection.

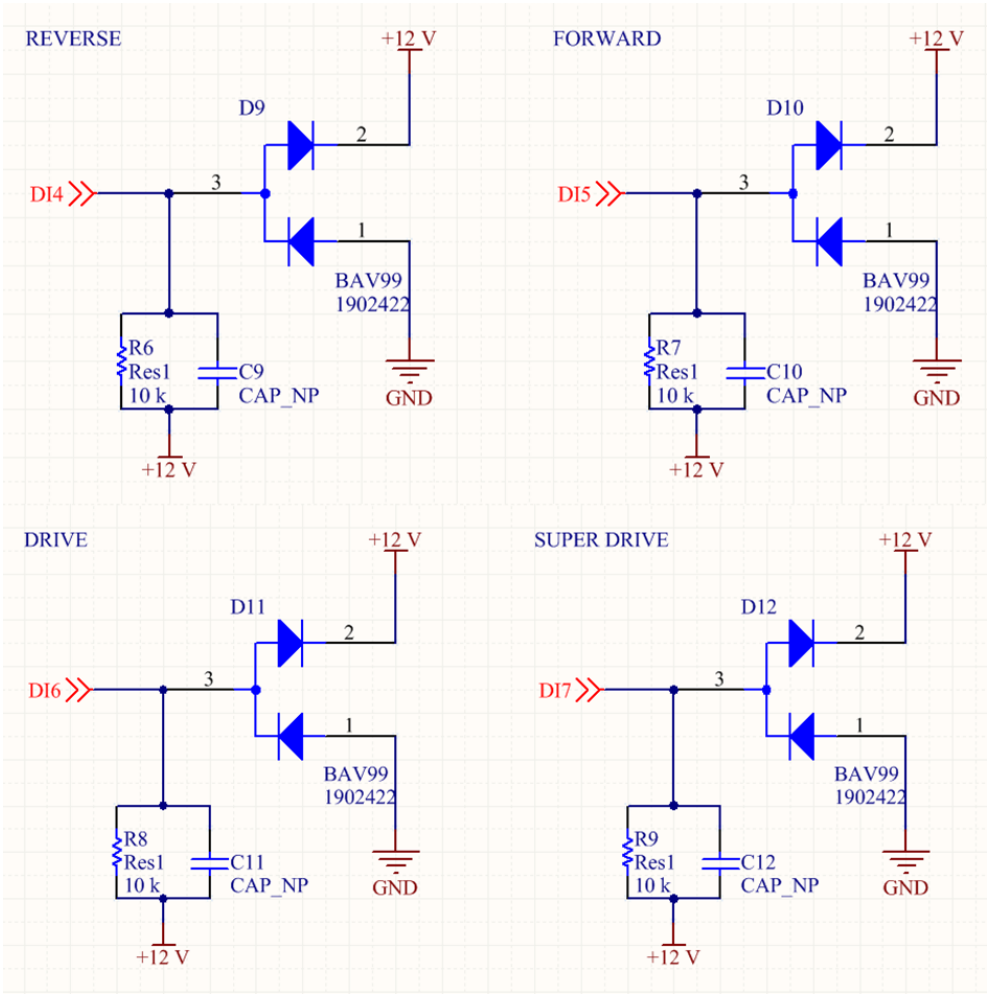


Figure 4.11 Lever position signals

4.4.5.3 x-IMU UART to RS232 converter

In order to connect the *x-IMU* to a SERIAL-CAN interface, a level conversion circuit was developed based on the integrated circuit. The connections are presented in Figure 4.12 and Figure 4.13.

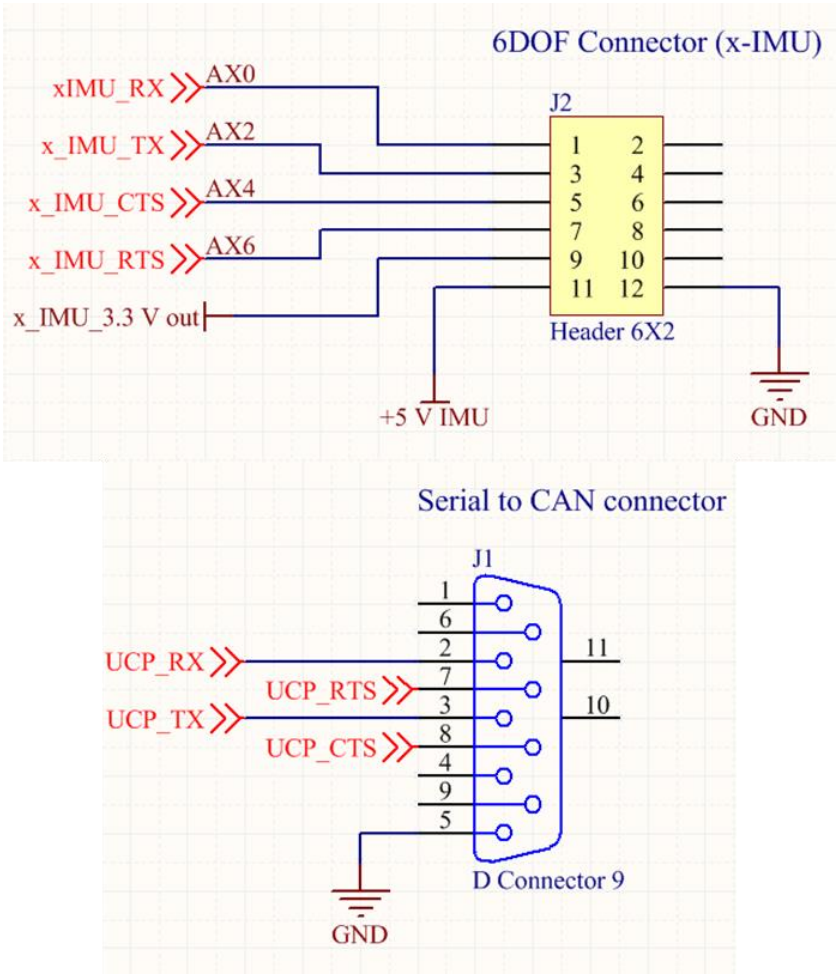


Figure 4.12 UART to SERIAL level converter connections

In accordance with the *x-IMU* and the *MAX3232E* specifications, the following electronic circuit was developed and tested, as shown in Paragraph 4.3.361.

Connections with the *x-IMU* are marked with a label starting with "x_IMU_...", while the "UCP_..." labeled ports are actually connected to the SERIAL-to-CAN interface.

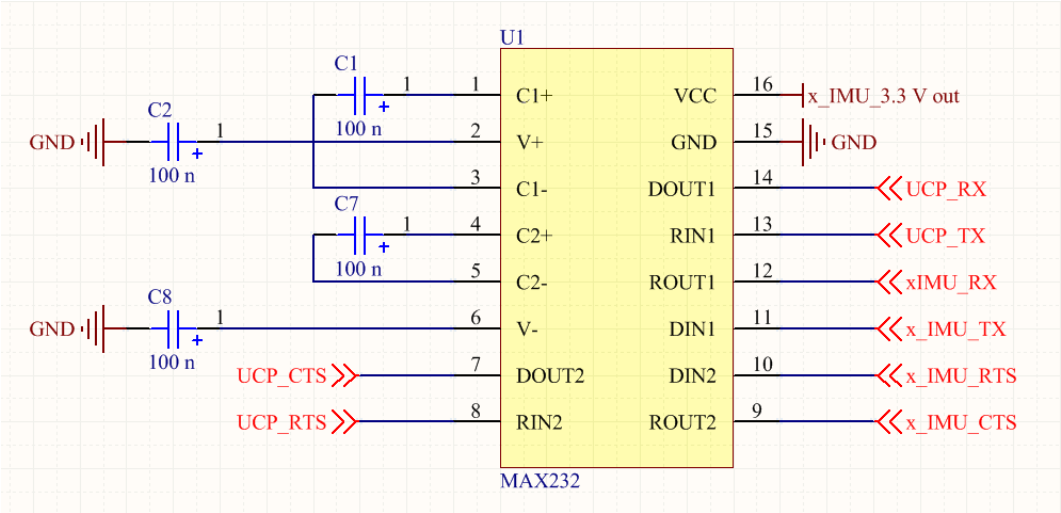


Figure 4.13 UART to SERIAL level converter integrated circuit

Summarizing, Figure 4.14 shows the acquisition chain developed for the Inertial Measurement Unit, as described in Paragraph 4.3.3 and 4.6.3.2)

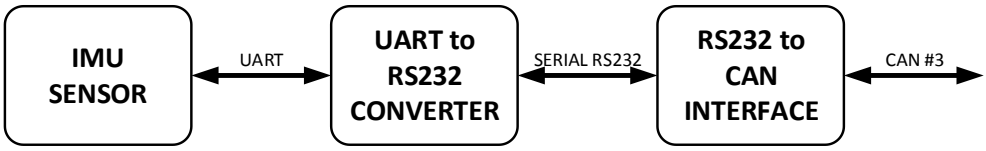


Figure 4.14 IMU device acquisition chain

4.4.6 CAN Networks

As previously introduced, a field bus network was deployed on the vehicle prototype, allowing the correct communication between the various installed ECUs. Four CAN networks were set up as outlined in Table 4.1. and Figure 4.15.

CAN network identification	UCP port connection	Baud Rate	CAN Standard	Field devices connection
CAN#1	Module 1 Port 1	1 Mbps	2.0A	BMS Front, BMS Rear, MFU
CAN#3	Module 2 Port 1	1 Mbps	2.0A	TCSS Front, TCSS Rear, IMU, Telemetry (optional)
CAN#4	Module 2 Port 0	1 Mbps	2.0A	Traction Drive Front, Traction Drive Rear
CAN#5	Module 1 Port 0	500 kbps	2.0B	Electric Power Steering (EPS)

Table 4.1 CAN network configuration

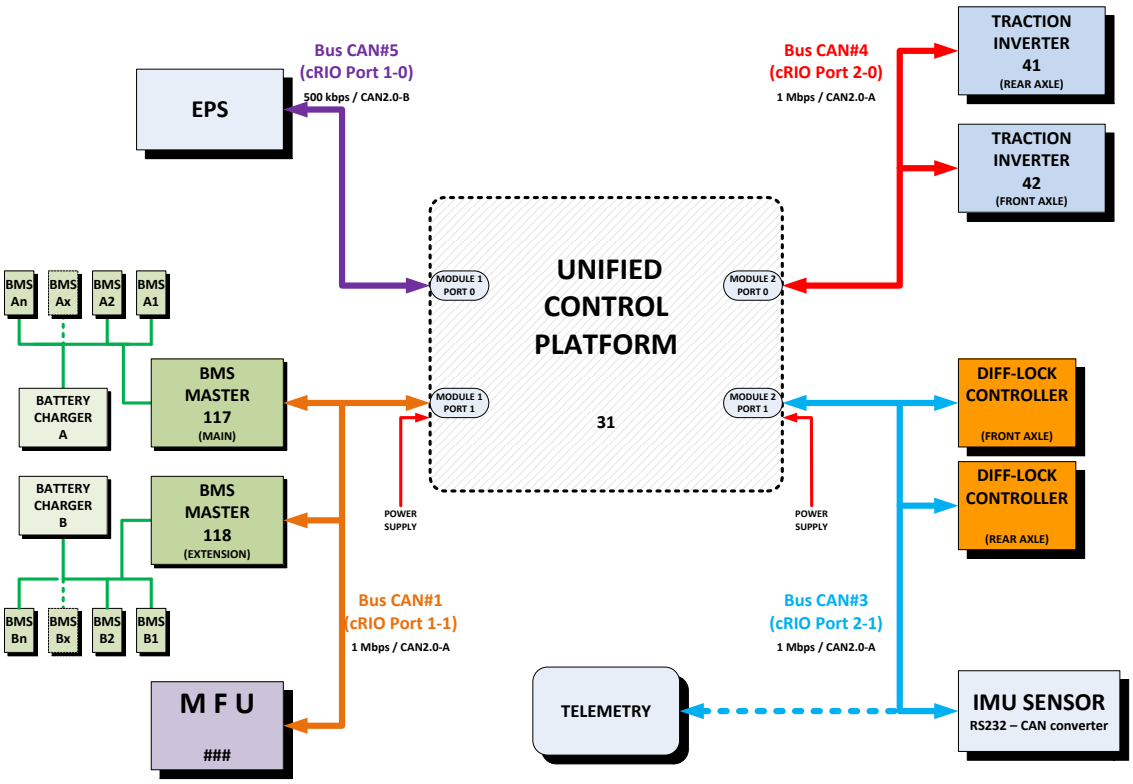


Figure 4.15 CAN network configuration

4.5 Signals description

Following the selection of the sensors to be used on the AMBER-ULV prototype, a collection of all the relevant information required for the design of the hardware interface between the *National Instruments* controller and field devices was carried out. The resulting document was then updated with the information necessary for the development of low-level UCP software.

In the following pages, a set of tables is presented, which are extracted from the document above described. Such a document, during the phases of UCP development, was also shared with the partners involved in tasks related to UCP, namely EOS, SHERPA and SIEMENS.

The document is related to those quantities that are “acquired” through the lower hardware level of the UCP controller, namely the FPGA. A similar report was prepared for the CAN bus broadcasted quantities, but it is not described in this manuscript.

The document is organized in a way that reflects the information flow.

The first part, Table 4.2, contains the sensor information, then the manufacturer name and the model name, the supply voltage, the reference to the datasheet and a brief description of the sensor working principle.

In Table 4.3 the information related to the sensor connection with FPGA is reported. The table contains the CompactRIO C-series module type and its position in the *cRIO* chassis, the channel name and the pin number, and the signal connection mode for each signal.

From Table 4.4 onwards, information concerning the UCP software is collected. Table 4.4 in particular, contains the lowest level acquisition settings, such as the channel operating range, the channel resolution, the loop name wherein the acquisition takes place and the sample rate.

It is important to consider that all the analog acquisitions are handled, on the FPGA target, with a fixed-point representation.

Most of the signals are also processed on the FPGA. This include filtering, but also conversions and validity checks. These operations are described in Table 4.5 and they are also the lasts performed by the FPGA before sending them to the real-time part. Table 4.5 also reports the variables through which the information is sent to the RT code.

Table 4.6 describes how the variables are imported into the real-time part of the UCP, reporting the loop name and its timing along with whether or not any conversion is applied and the data type the variable assumes within the RT code.

Table 4.7, summarizes the scaling settings, if implemented, for each signal acquired.

Table 4.8, finally, lists the global variables in which the processed information is available for the RT functions.

ACQUISITION						
	SENSOR					
NAME	MANU FACTURER	MODEL	SUPPLY VOLT.	SUPPLY SOURCE	Datasheet	brief description
gear selector SUPER_DRIVE	START	SCE07	12V	12V_UCP	Annex 10.2	From lever selector, Hall sensor. Open collector output. Active low
gear selector DRIVE	START	SCE07	12V	12V_UCP	Annex 10.2	From lever selector, Hall sensor. Open collector output. Active low
gear selector NEUTRAL	START	SCE07	12V	12V_UCP	Annex 10.2	From lever selector, Hall sensor. Open collector output. Active low
gear selector REVERSE	START	SCE07	12V	12V_UCP	Annex 10.2	From lever selector, Hall sensor. Open collector output. Active low
Gas Pedal Position	START	STL1D	5 V	12V_UCP	Annex 10.1	Hall effect, compensated
Gas Pedal Validation	START	STL1D	12V	12V_UCP	Annex 10.1	0=pedal released or at the beginning of the downward pedal stroke
Brake Pedal Position	START	STL1D	12V	12V_UCP	Annex 10.1	Hall effect, compensated
Brake Pedal Validation	START	STL1D	12V	12V_UCP	Annex 10.1	0=pedal released or at the beginning of the downward pedal stroke
F-L Wheel Speed Sensor	OMRON	E2A-S08KS02-WP-C1	12V	12V_UCP	Annex 10.6	Proximity
F-R Wheel Speed Sensor	OMRON	E2A-S08KS02-WP-C1	12V	12V_UCP	Annex 10.6	Proximity
R-L Wheel Speed Sensor	OMRON	E2A-S08KS02-WP-C1	12V	12V_UCP	Annex 10.6	Proximity
F-R Wheel Speed Sensor	OMRON	E2A-S08KS02-WP-C1	12V	12V_UCP	Annex 10.6	Proximity
BRAKE PRESSURE 1	AVIO RACE SOLUTIONS	SP100_M10x1	5V	5V_UCP BOX	Annex 10.5	Active, ratiometric
BRAKE PRESSURE 2	AVIO RACE SOLUTIONS	SP100_M10x1	5V	5V_UCP BOX	Annex 10.5	Active, ratiometric
Front motor temperature	GE	DKF103N3	5V	5V_UCP BOX	Annex 10.3	NTC resistor installed inside the motor. Voltage divider required
Rear motor temperature	GE	DKF103N3	5V	5V_UCP BOX	Annex 10.3	NTC resistor installed inside the motor. Voltage divider required
Front TCSS temperature	VDO	323-801-017-001K/N	5V	5V_UCP BOX	Annex 10.4	NTC resistor in close contact with oil. Voltage divider required
Rear TCSS temperature	VDO	323-801-017-001K/N	5V	5V_UCP BOX	Annex 10.4	NTC resistor in close contact with oil. Voltage divider required

Table 4.2 Sensor physical information

	FPGA DOMAIN				
	PHISICAL CONNECTION				
NAME	Module type	MODULE POSITION	Channel Id	pin	Terminal Mode
gear selector SUPER_DRIVE	NI9425	5	DI7	8	Sinking Digital
gear selector DRIVE	NI9425	5	DI6	7	Sinking Digital
gear selector NEUTRAL	NI9425	5	DI5	6	Sinking Digital
gear selector REVERSE	NI9425	5	DI4	5	Sinking Digital
Gas Pedal Position	NI9205	3	AI1	2	Single Ended
Gas Pedal Validation	NI9425	5	DI1	2	Sinking Digital
Brake Pedal Position	NI9205	3	AI2	3	Single Ended
Brake Pedal Validation	NI9425	5	DI2	3	Sinking Digital
F-L Wheel Speed Sensor	NI9411	4	DI0a	1	Single Ended
F-R Wheel Speed Sensor	NI9411	4	DI1a	2	Single Ended
R-L Wheel Speed Sensor	NI9411	4	DI2a	3	Single Ended
F-R Wheel Speed Sensor	NI9411	4	DI3a	6	Single Ended
BRAKE PRESSURE 1	NI9205	4	AI3	4	Single Ended
BRAKE PRESSURE 2	NI9205	4	AI4	5	Single Ended
Front motor temperature	NI9205	3	AI8	20	Single Ended
Rear motor temperature	NI9205	3	AI9	21	Single Ended
Front TCSS temperature	NI9205	3	AI10	22	Single Ended
Rear TCSS temperature	NI9205	3	AI11	23	Single Ended

Table 4.3 Connection to FPGA

	FPGA DOMAIN							
	SAMPLING & CONVERSION							
NAME	Channel Range	Res.	DATA Type	SIGN	WORD LENGTH	INTEGER WORD LENGHT	FPGA Loop Reference	SAMPLE RATE [ms]
gear selector SUPER_DRIVE	0-24V	/	boolean	/	/	/	Gear Selector Acquisition	5
gear selector DRIVE	0-24V	/	boolean	/	/	/	Gear Selector Acquisition	5
gear selector NEUTRAL	0-24V	/	boolean	/	/	/	Gear Selector Acquisition	5
gear selector REVERSE	0-24V	/	boolean	/	/	/	Gear Selector Acquisition	5
Gas Pedal Position	+/-10V	16bit	Fixed point	signed	16	5	GAS Pedal Position & Validation; BRAKE Pedal Position & Validation	2
Gas Pedal Validation	0-24V	/	boolean	/	/	/	GAS Pedal Position & Validation; BRAKE Pedal Position & Validation	2
Brake Pedal Position	+/-10V	16bit	Fixed point	signed	16	5	GAS Pedal Position & Validation; BRAKE Pedal Position & Validation	2
Brake Pedal Validation	0-24V	/	boolean	/	/	/	GAS Pedal Position & Validation; BRAKE Pedal Position & Validation	2
F-L Wheel Speed Sensor	0-24V	/	boolean	/	/	/	Phonic Wheels Period-Meter	0.005
F-R Wheel Speed Sensor	0-24V	/	boolean	/	/	/	Phonic Wheels Period-Meter	0.005
R-L Wheel Speed Sensor	0-24V	/	boolean	/	/	/	Phonic Wheels Period-Meter	0.005
F-R Wheel Speed Sensor	0-24V	/	boolean	/	/	/	Phonic Wheels Period-Meter	0.005
BRAKE PRESSURE 1	+/-10V	12bit	Fixed point	±	16	5	FL-RR Brake Pressure; FR-RL Brake Pressure	10
BRAKE PRESSURE 2	+/-10V	12bit	Fixed point	±	16	5	FL-RR Brake Pressure; FR-RL Brake Pressure	10
Front motor temperature	+/-10V	12bit	Fixed point	±	16	5	Motor 1 Temp.; Motor 2 Temp.; AUX 1 Temp.; AUX 2 Temp.	250
Rear motor temperature	+/-10V	12bit	Fixed point	±	16	5	Motor 1 Temp.; Motor 2 Temp.; AUX 1 Temp.; AUX 2 Temp.	250
Front TCSS temperature	+/-10V	12bit	Fixed point	±	16	5	Motor 1 Temp.; Motor 2 Temp.; AUX 1 Temp.; AUX 2 Temp.	250
Rear TCSS temperature	+/-10V	12bit	Fixed point	±	16	5	Motor 1 Temp.; Motor 2 Temp.; AUX 1 Temp.; AUX 2 Temp.	250

Table 4.4 FPGA acquisition information

ACQUISITION	FPGA DOMAIN					FPGA-RT INTERF.
	PRE-PROCESSING			VALIDITY CHECK		
NAME	Filter Type	Filter Characteristics	UPDATE RATE [ms]	VALID RANGE	Validity Check Variable	variable name
gear selector SUPER_DRIVE	truth table conversion	7.2.2	5	0 or 1	Table 7.1	Gear Sel Pos
gear selector DRIVE	truth table conversion	7.2.2	5	0 or 1	Table 7.1	Gear Sel Pos
gear selector NEUTRAL	truth table conversion	7.2.2	5	0 or 1	Table 7.1	Gear Sel Pos
gear selector REVERSE	truth table conversion	7.2.2	5	0 or 1	Table 7.1	Gear Sel Pos
Gas Pedal Position	Multichannel Butterwoth Filter	Lowpass, 2nd order, rate 500Sa/s, cutoff f 1/5	2	tbd	**AI Pedal Range** element 0	**AI Pedal** element 0
Gas Pedal Validation	none	none	2	none	none	**Pedal Valitadion** element 0
Brake Pedal Position	Multichannel Butterwoth Filter	Lowpass, 2nd order, rate 500Sa/s, cutoff f 1/5	2	tbd	**AI Pedal Range** element 1	**AI Pedal** element 1
Brake Pedal Validation	none	none	2	none	none	**Pedal Valitadion** element 1
F-L Wheel Speed Sensor	period-meter	multiple of 5 µs	10	check on update within period	**Wheel Update Count** element 0	**Wheel Period** element 0
F-R Wheel Speed Sensor	period-meter	multiple of 5 µs	10	check on update within period	**Wheel Update Count** element 1	**Wheel Period** element 1
R-L Wheel Speed Sensor	period-meter	multiple of 5 µs	10	check on update within period	**Wheel Update Count** element 2	**Wheel Period** element 2
F-R Wheel Speed Sensor	period-meter	multiple of 5 µs	10	check on update within period	**Wheel Update Count** element 3	**Wheel Period** element 3
BRAKE PRESSURE 1	Multichannel Butterwoth Filter	Lowpass, 2nd order, rate 100Sa/s, cutoff f 1/10	10	tbd	**AI Press Range** element 0	**AI Press** element 0
BRAKE PRESSURE 2	Multichannel Butterwoth Filter	Lowpass, 2nd order, rate 100Sa/s, cutoff f 1/10	10	tbd	**AI Press Range** element 1	**AI Press** element 1
Front motor temperature	Multichannel Butterwoth Filter	Lowpass, 2nd order, rate 4Sa/s, cutoff f 1/4	250	tbd	**AI Temps Range** element 0	**AI Temps** element 0
Rear motor temperature	Multichannel Butterwoth Filter	Lowpass, 2nd order, rate 4Sa/s, cutoff f 1/4	250	tbd	**AI Temps Range** element 1	**AI Temps** element 1
Front TCSS temperature	Multichannel Butterwoth Filter	Lowpass, 2nd order, rate 4Sa/s, cutoff f 1/4	250	tbd	**AI Temps Range** element 2	**AI Temps** element 2
Rear TCSS temperature	Multichannel Butterwoth Filter	Lowpass, 2nd order, rate 4Sa/s, cutoff f 1/4	250	tbd	**AI Temps Range** element 3	**AI Temps** element 3

Table 4.5 FPGA signal processing and interface with RT

ACQUISITION	REAL TIME			
	INPUT LOOP		DATA TYPE	
NAME	NAME	PERIOD [ms]	TYPE CONVERSION	INPUT DATA TYPE
gear selector SUPER_DRIVE	RLTM_Intrf_Lever_v17	FPGA sync IRQ5	no	uint8
gear selector DRIVE	RLTM_Intrf_Lever_v17	FPGA sync IRQ5	no	uint8
gear selector NEUTRAL	RLTM_Intrf_Lever_v17	FPGA sync IRQ5	no	uint8
gear selector REVERSE	RLTM_Intrf_Lever_v17	FPGA sync IRQ5	no	uint8
Gas Pedal Position	RLTM_Intrf_Pedals_v17	FPGA sync IRQ3	fxp to dbl	fxp<±,16,5>
Gas Pedal Validation	RLTM_Intrf_Pedals_v17	FPGA sync IRQ3	boolean to 0-1 + to dbl	boolean
Brake Pedal Position	RLTM_Intrf_Pedals_v17	FPGA sync IRQ3	fxp to dbl	fxp<±,16,5>
Brake Pedal Validation	RLTM_Intrf_Pedals_v17	FPGA sync IRQ3	boolean to 0-1 + to dbl	boolean
F-L Wheel Speed Sensor	RLTM_Intrf_WhlSpd_v17	FPGA sync IRQ4	uint16 to dbl	uint16
F-R Wheel Speed Sensor	RLTM_Intrf_WhlSpd_v17	FPGA sync IRQ4	uint16 to dbl	uint16
R-L Wheel Speed Sensor	RLTM_Intrf_WhlSpd_v17	FPGA sync IRQ4	uint16 to dbl	uint16
F-R Wheel Speed Sensor	RLTM_Intrf_WhlSpd_v17	FPGA sync IRQ4	uint16 to dbl	uint16
BRAKE PRESSURE 1	RLTM_Intrf_Press_v17	FPGA sync IRQ2	fxp to dbl	fxp<±,16,5>
BRAKE PRESSURE 2	RLTM_Intrf_Press_v17	FPGA sync IRQ2	fxp to dbl	fxp<±,16,5>
Front motor temperature	RLTM_Intrf_Temps_v17	100	fxp to dbl	fxp<±,16,5>
Rear motor temperature	RLTM_Intrf_Temps_v17	100	fxp to dbl	fxp<±,16,5>
Front TCSS temperature	RLTM_Intrf_Temps_v17	100	fxp to dbl	fxp<±,16,5>
Rear TCSS temperature	RLTM_Intrf_Temps_v17	100	fxp to dbl	fxp<±,16,5>

Table 4.6 Real Time data interface

ACQUISITION	REAL TIME						
	SCALING						
NAME	Scaling Loop	Scaling Period	Measurement unit	Scale Factor	MIN value	MAX value	Offset
gear selector SUPER_DRIVE	NA	NA	NA	NO	NA	NA	NO
gear selector DRIVE	NA	NA	NA	NO	NA	NA	NO
gear selector NEUTRAL	NA	NA	NA	NO	NA	NA	NO
gear selector REVERSE	NA	NA	NA	NO	NA	NA	NO
Gas Pedal Position	same as input	same as input	p.u.	1/5	0	1	NO
Gas Pedal Validation	same as input	same as input	boolean	NO	0	1	NO
Brake Pedal Position	same as input	same as input	p.u	1/5	0	1	NO
Brake Pedal Validation	same as input	same as input	boolean	NO	0	1	NO
F-L Wheel Speed Sensor	same as input	same as input	[rad/s]	27925.2680	0	130	NO
F-R Wheel Speed Sensor	same as input	same as input	[rad/s]	27925.2680	0	130	NO
R-L Wheel Speed Sensor	same as input	same as input	[rad/s]	27925.2680	0	130	NO
F-R Wheel Speed Sensor	same as input	same as input	[rad/s]	27925.2680	0	130	NO
BRAKE PRESSURE 1	same as input	same as input	[bar]	25.0000	0	100	-12.5
BRAKE PRESSURE 2	same as input	same as input	[bar]	25.0000	0	100	-12.5
Front motor temperature	same as input	same as input	[°C]	7.2.3	-30	200	7.2.3
Rear motor temperature	same as input	same as input	[°C]	7.2.3	-30	200	7.2.3
Front TCSS temperature	same as input	same as input	[°C]	7.2.3	-30	200	7.2.3
Rear TCSS temperature	same as input	same as input	[°C]	7.2.3	-30	200	7.2.3

Table 4.7 Real Time data processing information

ACQUISITION	REAL TIME		
	OUTPUT		
NAME	OUT DATA TYPE	UPDATE RATE [ms]	VARIABLE NAME
gear selector SUPER_DRIVE	double	same as input	GV_LeverPos
gear selector DRIVE	double	same as input	GV_LeverPos
gear selector NEUTRAL	double	same as input	GV_LeverPos
gear selector REVERSE	double	same as input	GV_LeverPos
Gas Pedal Position	double	same as input	GV_Pedals, Element "Gas Signal"
Gas Pedal Validation	double	same as input	GV_Pedals, Element "Gas Valid"
Brake Pedal Position	double	same as input	GV_Pedals, Element "Brk Signal"
Brake Pedal Validation	double	same as input	GV_Pedals, Element "Brk Valid"
F-L Wheel Speed Sensor	double	same as input	GV_WheelSpeed, Element "FL Speed"
F-R Wheel Speed Sensor	double	same as input	GV_WheelSpeed, Element "FR Speed"
R-L Wheel Speed Sensor	double	same as input	GV_WheelSpeed, Element "RL Speed"
F-R Wheel Speed Sensor	double	same as input	GV_WheelSpeed, Element "RR Speed"
BRAKE PRESSURE 1	double	same as input	GV_BrakePress, Element "Press Brk1"
BRAKE PRESSURE 2	double	same as input	GV_BrakePress, Element "Press Brk2"
Front motor temperature	double	same as input	GV_Temps, Element "Temp Mot2"
Rear motor temperature	double	same as input	GV_Temps, Element "Temp Mot1"
Front TCSS temperature	double	same as input	GV_Temps, Element "Temp Aux2"
Rear TCSS temperature	double	same as input	GV_Temps, Element "Temp Aux1"

Table 4.8 Real Time global variables

4.6 UCP Low Level Software

4.6.1 General approach for UCP algorithm

Part of the Ph.D. activity consisted in the development of the low-level algorithms for the correct management of the field devices, sensors and actuators.

This activity required the development of two *LabVIEW* software, one for the FPGA chip, which is then automatically translated to *HDL* programming language thanks to a dedicated tool, and the other one for the real-time controller. The capability to exchange information between them was also implemented.

Figure 4.17 reports the scheme of the implemented architecture. As it can be seen, the I/O modules are connected to the FPGA. Dedicated functions like filtering, range validity checks, "FIFO" buffers and other necessary computations were developed for the FPGA target of the UCP. Additionally, actuator control loops were developed, such as the CAN frame transmissions, digital outputs with PWM functions and analog outputs. The real-time microcontroller, as it exchanges data with FPGA, receives the acquired or received information and sends the reference values for output controls.

Figure 4.16 provides a legend of how the information exchange is managed from the FPGA to the Real-Time controller of the UCP and vice-versa.

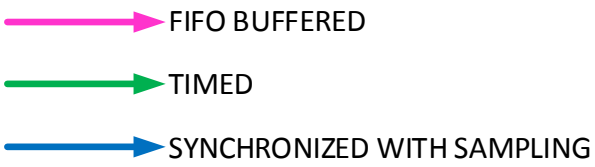


Figure 4.16 Data sharing strategies used in the UCP algorithm

As it is shown, three different solutions are adopted, depending on the criticality and the requested update rate of the data.

As an example, the CAN frames from the BMS system and the traction drives are sent to the RT processor as soon as they are received, by means of a FIFO buffer, in order to reduce the amount of calculations on the FPGA deriving from rescaling [26].

Considering specific input variables, such as the gas and brake pedal positions, the acquiring and processing loop on the FPGA sets an interrupt every time a new data is available to which the RT controller synchronizes its interface loop with, in order to keep the total delay as low as possible.

Eventually, for those variables that are associated to physical quantities with low dynamics, simple timed loops are used.

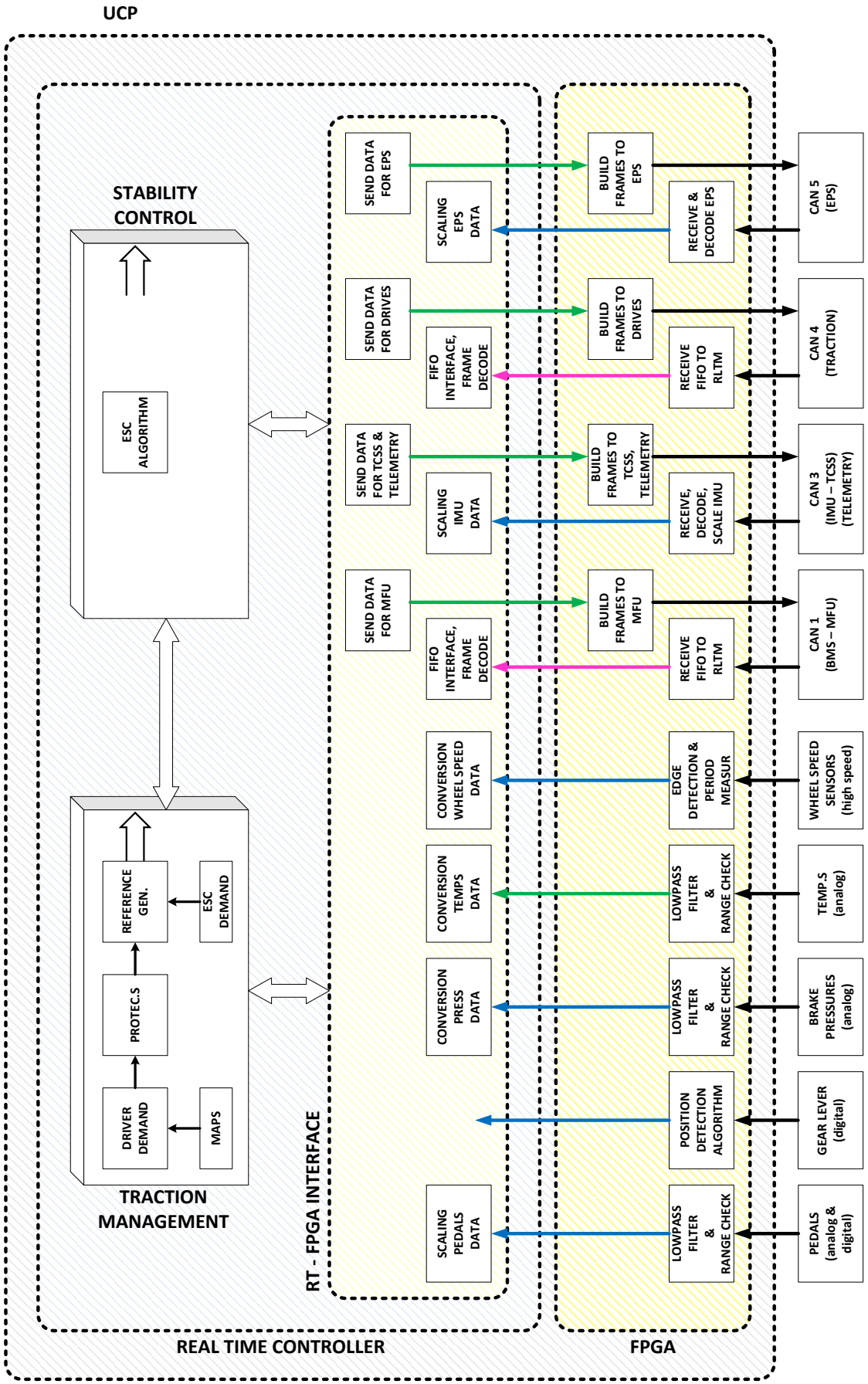


Figure 4.17 UCP software architecture scheme

Figure 4.18 shows what the software developed (the real-time part) looks like in the LabVIEW environment.

The lower part of the code snippet is relative to the traction management and stability control algorithms, while the whole upper part is formed by the low-level functions, which contains, in addition to the communication with the FPGA, all the scaling, decoding and filtering functions required.

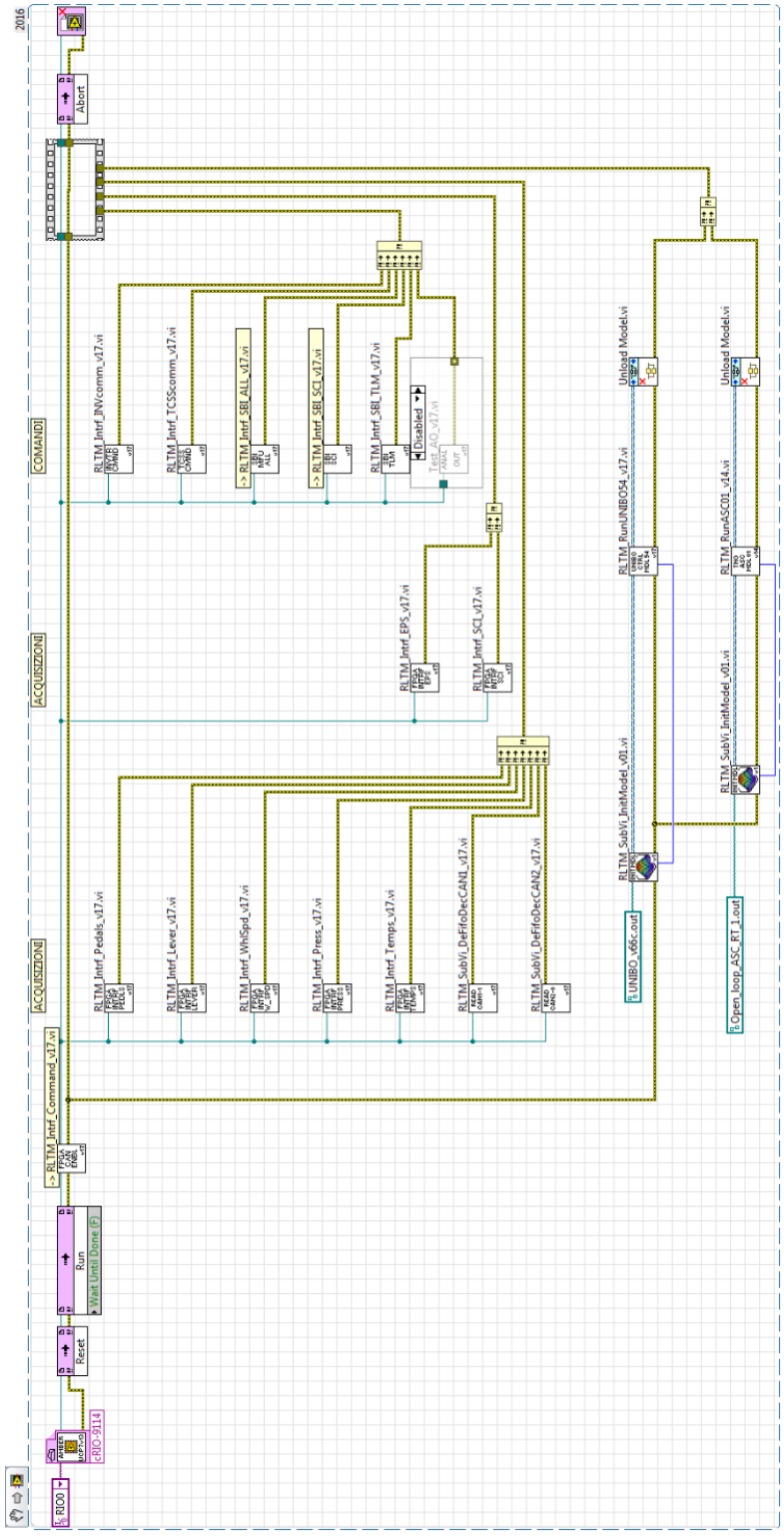


Figure 4.18 Real Time software

In the following, some examples are reported with the purpose to show the general coding strategy used in the development of the UCP software.

4.6.2 Low Level I/O functions

4.6.2.1 Gas and Brake pedal acquisition

Figure 4.19 and Figure 4.20 show the implementation of the Gas and Brake pedal acquisition for the FPGA and the real-time controller respectively. The two pedals are managed together in the same loop both on the FPGA and RT side. In the FPGA loop it can be seen how both the analog signals and digital signals (see 4.2.2.1) are acquired simultaneously, with a sampling period of 2 ms. Moreover, for the analog signals a 2nd order *Butterworth* filter is applied, in order to reduce noise. A range validity check is also performed with the purpose of increasing the detection probability of hardware failures in the acquisition chain.

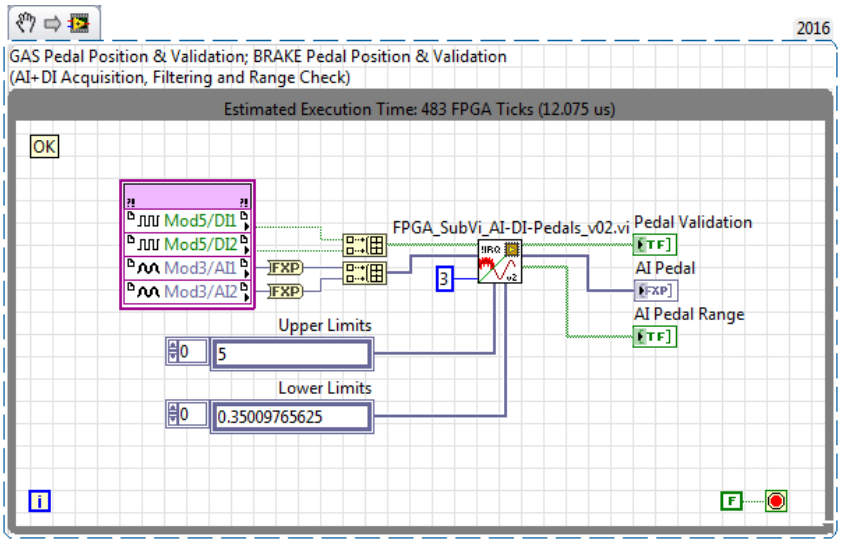


Figure 4.19 FPGA loop for Gas and Brake pedal acquisition

After the acquisition and signal processing loop has finished, it sets an interrupt the RT synchronizes with, as it can be seen in Figure 4.20. In the RT loop the signal is scaled to [pu.] dividing it by the maximum output of the sensor, in this case 5 V.

After the processing described above, the variables are addressed into a *global variable* and are then available for any functions that need this information within the RT software side.

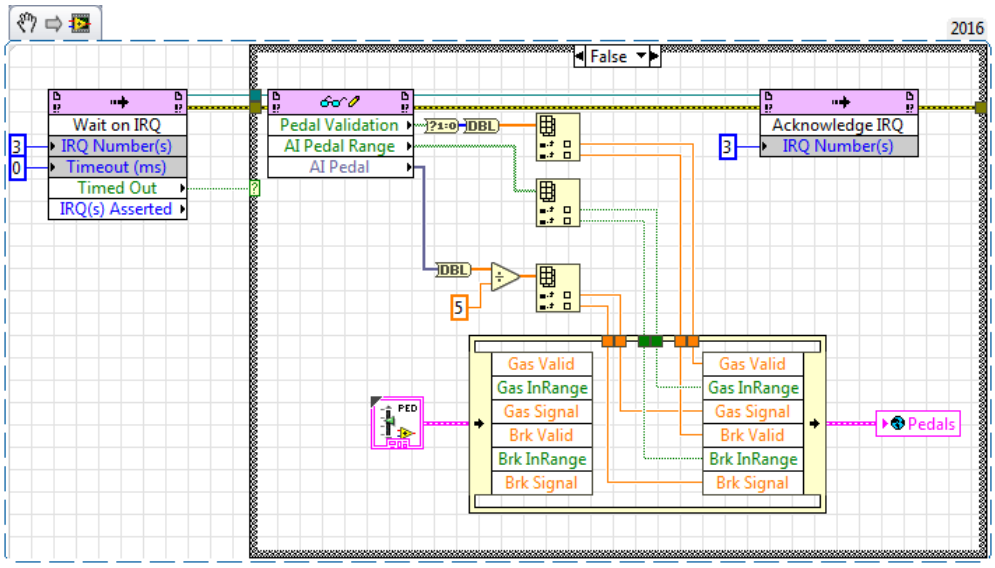


Figure 4.20 Real Time loop for Gas and Brake pedal acquisition

4.6.2.2 Direction lever acquisition

Figure 4.21 shows the FPGA loop in charge of acquiring and computing of the direction lever position. Although the sampling frequency of the four digital signals is set to 500 Sa/s, the acquisition loop is designed so that a transition in the lever position is trusted only if the signals remain stable within a time window of 50 ms, in order to discard any possible spike, thus preventing unexpected vehicle behaviour.

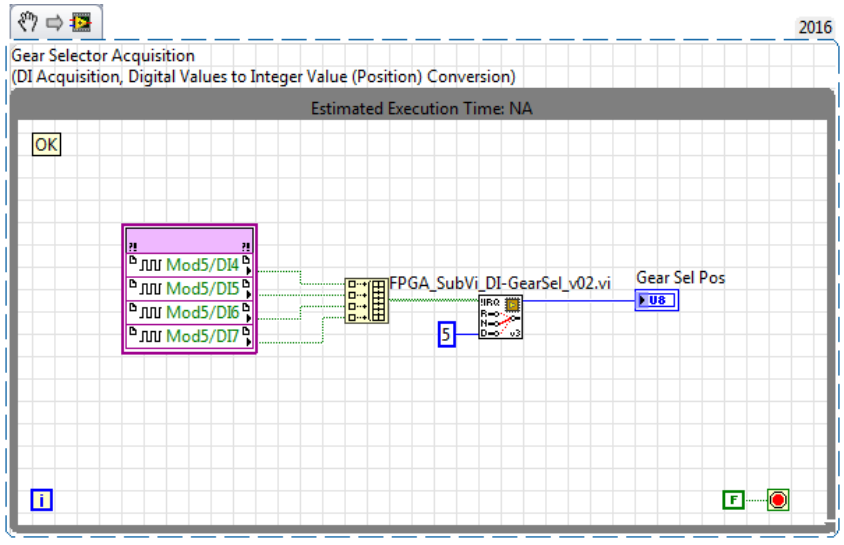


Figure 4.21 FPGA loop for direction lever acquisition

The digital signals are then converted into a position value according to the truth table presented in the following Table 4.9.

Unexpected values (which may be caused by a hardware failure) are assigned as “GEAR ERROR”, and this state is safely handled by the traction management algorithms in the RT target.

Lever Position	"DI4" value	"DI5" value	"DI6" value	"DI7" value	*Boolean array to number*	Output
REVERSE	1	0	0	0	1	1
NEUTRAL	0	0	0	0	0	2
DRIVE	0	1	1	0	6	3
SUPER DRIVE	0	1	0	1	10	4
GEAR ERROR	?	?	?	?	None of the previous	5

Table 4.9 Truth Table for the lever position

As for the Gas and Brake pedals, the lever position is also sent, as soon as it is available, to the real-time target with an interrupt synchronization mechanism.

As it can be seen in Figure 4.22, its value is then made available for the whole RT code through a global variable.

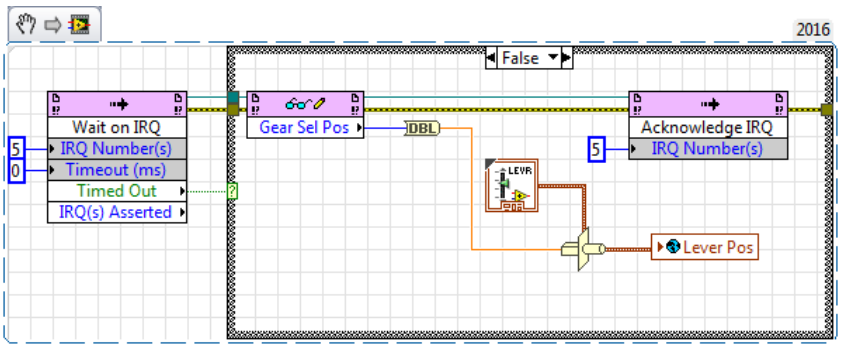


Figure 4.22 Real Time loop for direction lever acquisition

4.6.2.3 Temperature acquisition

To ensure safe operation of the prototype, it is necessary to measure the temperature of critical components, in order to trigger the necessary protections in case of abnormalities.

On the AMBER vehicle, some critical components, such as traction inverters and BMS, independently measure system temperatures and transmit them to the UCP via CAN-Bus. The two traction motors and actuators for the stability control (TCSS) are also equipped with temperature sensors but they are acquired directly by the UCP.

Figure 4.23 shows the acquisition loop implemented on the FPGA. It performs the acquisition with a sample period of 250 ms, the low-pass 2nd order *Butterworth* filter and the range check for two motor temperatures and two TCSS temperatures, simultaneously.

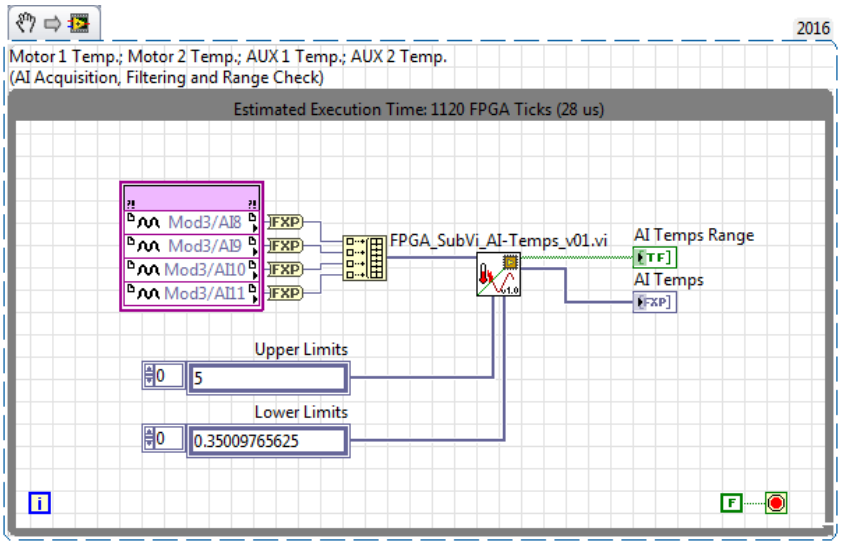


Figure 4.23 FPGA loop for motors and TCSSs temperatures acquisition

The acquired temperatures are then received by the RT loop shown in Figure 4.24, in which the acquired voltage is scaled to obtain degrees Celsius.

As described in the previous chapters, motor and TCSS temperatures are measured by means of thermistors, connected in voltage dividers. The voltage—temperature characteristics is therefore non-linear.

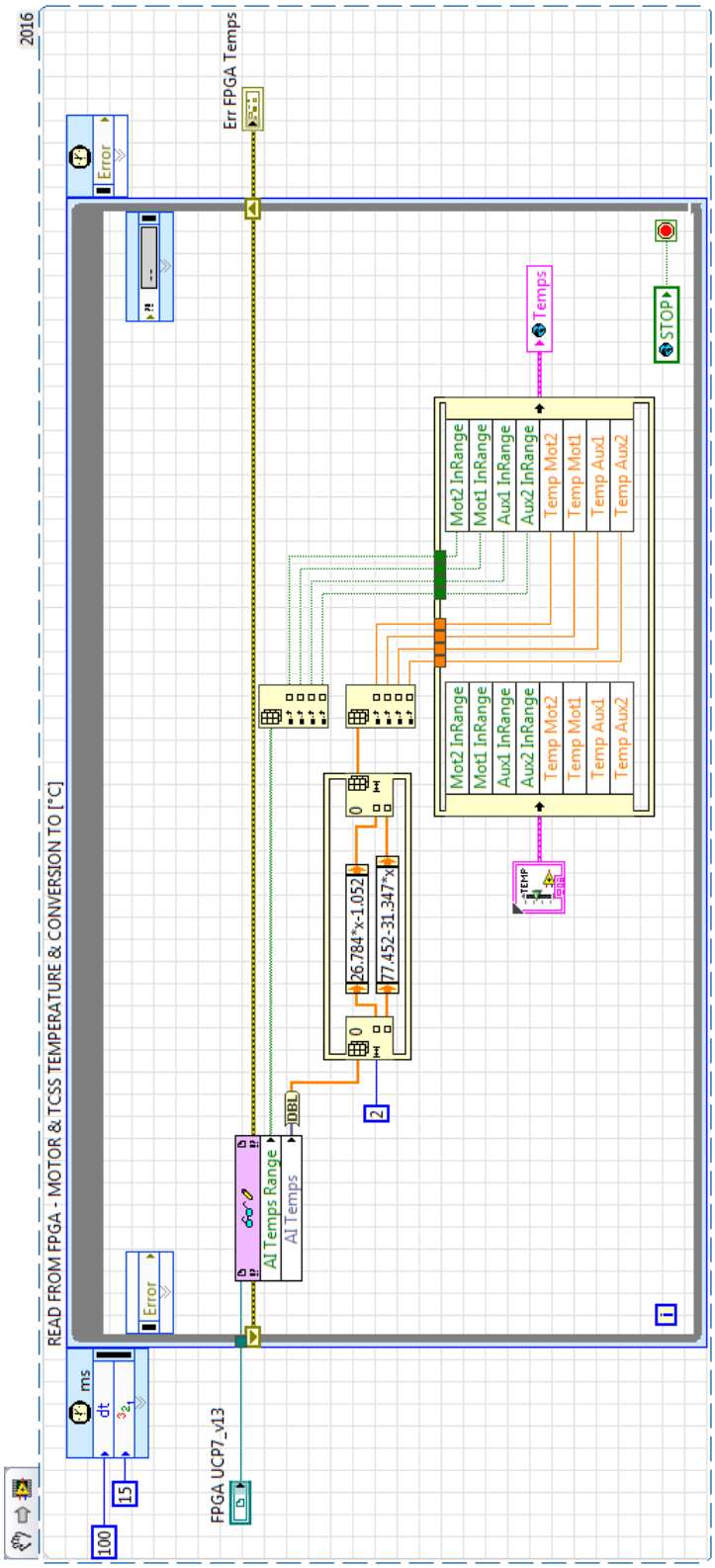


Figure 4.24 Real Time loop for motors and TCSSs temperatures acquisition

In order to simplify the conversion, the voltage—temperature relationship was linearized in the normal operating range, obtaining the trends shown in Figure 4.25 and Figure 4.26, for the measurement of motor and TCSS temperatures respectively.

A trend line is also shown in the figures, whose equation was used as conversion factor, as shown in Figure 4.24.

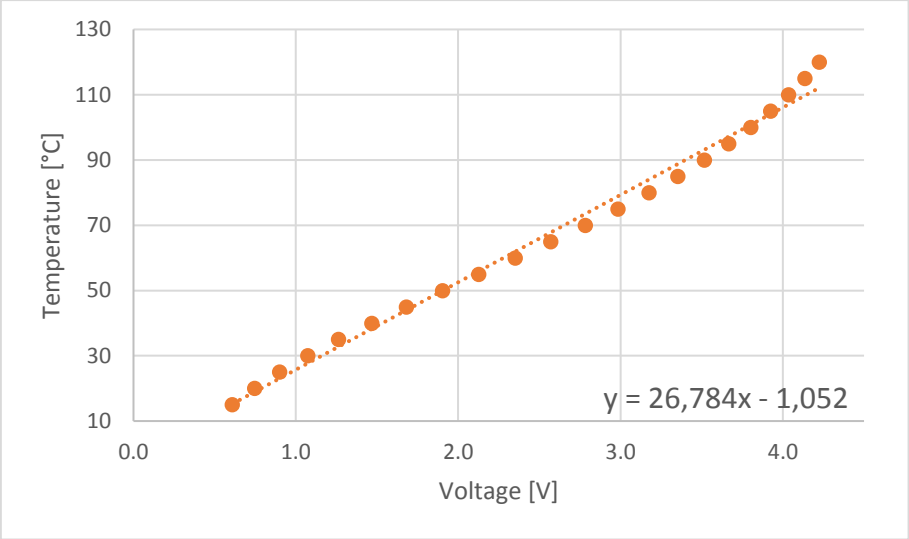


Figure 4.25 Linearized motor temperature sensor characteristics

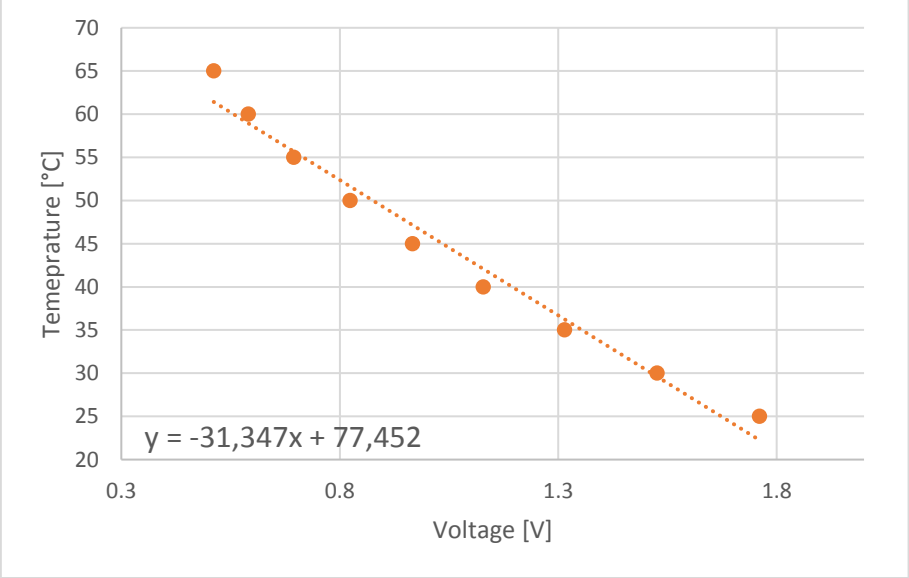


Figure 4.26 Linearized TCSS temperature sensor characteristics

4.6.2.4 Brake pressure acquisition

As described earlier, the hydraulic pressure of the two brake circuits is measured on the AMBER-ULV prototype. The two pressure sensors provide a ratiometric voltage signal proportional to the applied hydraulic pressure, acquired by the loop shown in Figure 4.27, in which the signals are also filtered and the validity range is checked.

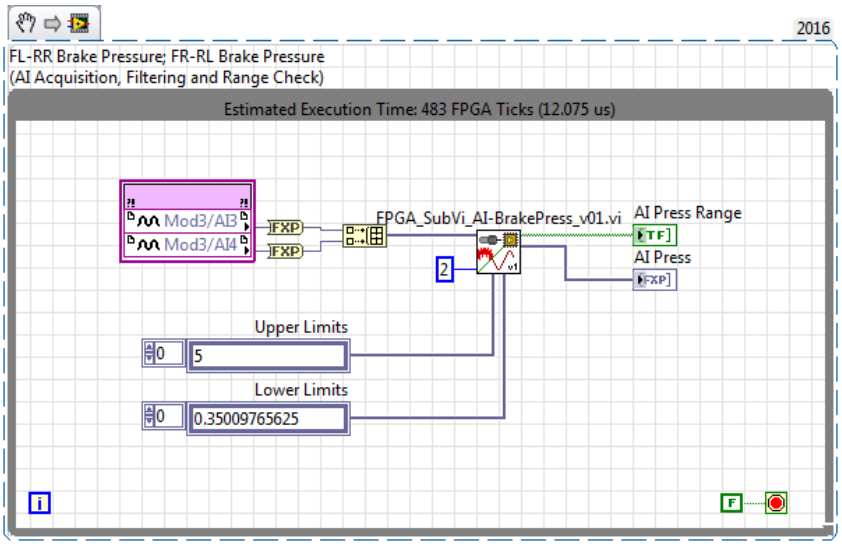


Figure 4.27 FPGA loop for brake pressure acquisition

The signals, acquired within a sampling period of 10 ms, are then sent to the RT loop, in which they are scaled according to the sensor specifications and made available in a dedicated global variable, as shown in Figure 4.28.

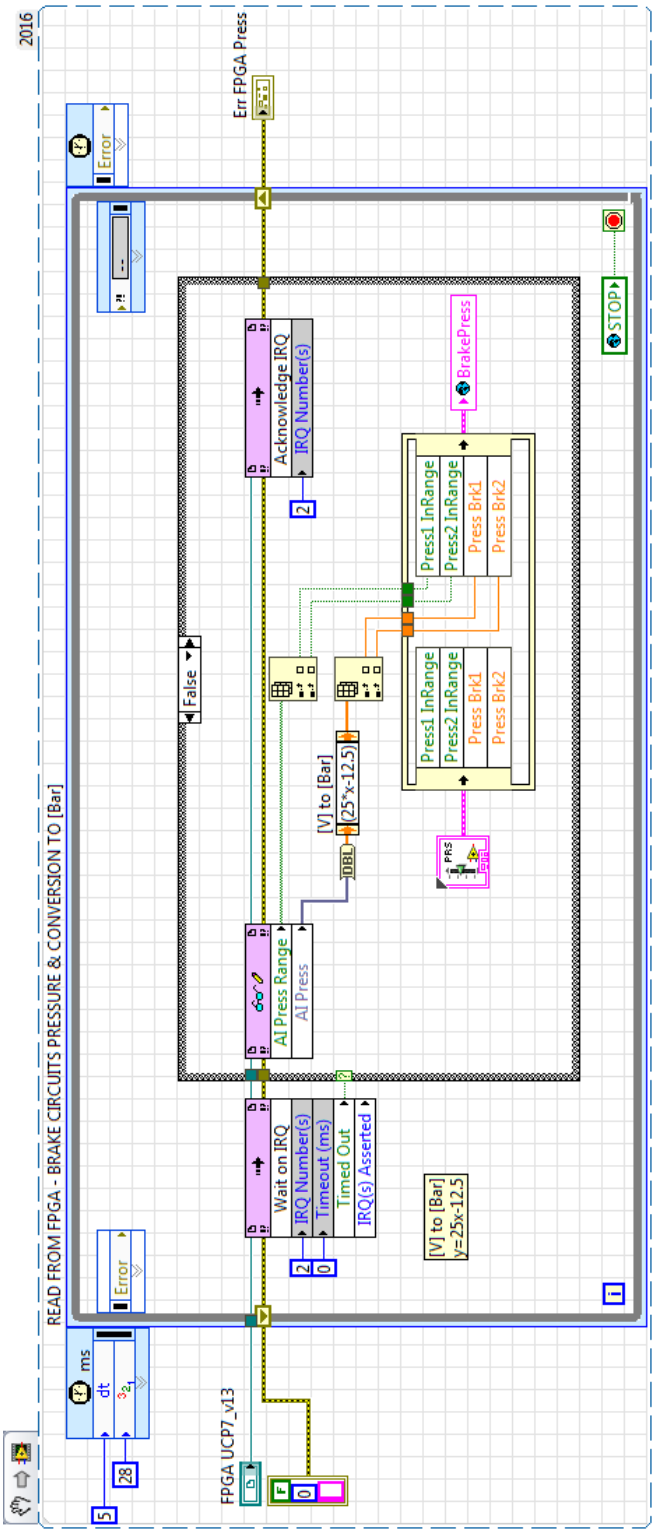


Figure 4.28 Real Time loop for brake pressure acquisition

4.6.2.5 Wheel speed acquisition

The individual wheel speed evaluation can be considered, as a more precise definition, an estimation, rather than a measurement.

As introduced in paragraph 4.2.2.6, in fact, the measurement principle consists of a proximity sensor facing a phonic wheel which in turn is rigidly coupled to the wheel hub. The proximity sensor changes its digital output state whether or not it is in front of a tooth of the phonic wheel.

With the wheel moving, the output signal assumes a square waveform (see Figure 4.4).

By measuring the waveform period, or its frequency, we get information about the wheel rotational speed.

On the FPGA software (Figure 4.29), the wheel speed is estimated through the period-meter algorithm. The period estimation is carried out for both rising and falling edges of the acquired signal. Moreover, a validity variable is used to notify that, at very low speeds, the update may not occur within a given time limit.

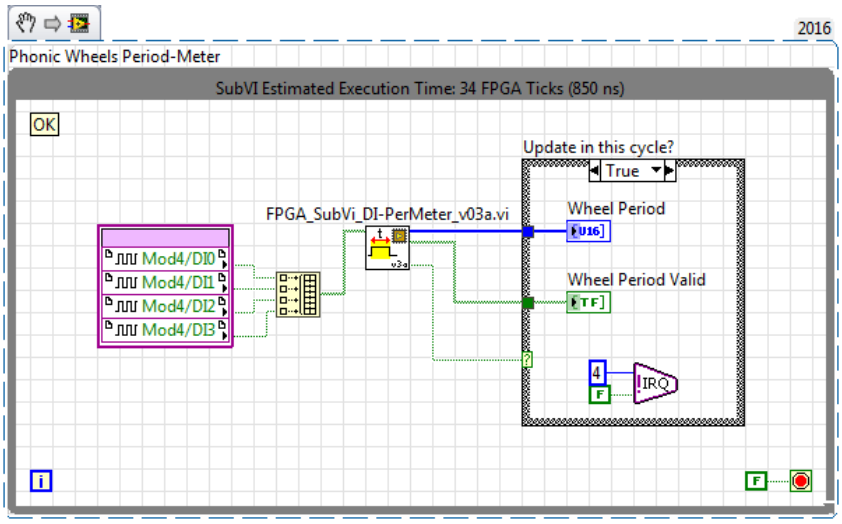


Figure 4.29 FPGA loop for wheel speed acquisition

The FPGA function calculates the signal period as a multiple of 5 µs, which is the sampling period of the digital signals coming from the proximity sensors. At each sampling a counter is incremented and its value is written on the *Wheel Period* variable each time an edge is detected.

The function running on the RT target, then, calculates the wheels speed through Equation 4-1, where ω_W is the calculated wheel speed in [rad/s], T_{Sa} is the sampling period in [s], z is the number of teeth of the phonic wheel and T_{Cnt} is the counter value between two signal edges of the same direction.

$$\omega_W = \frac{1}{T_{Sa}} \frac{2\pi}{z} \frac{1}{T_{Cnt}}$$

Equation 4-1

The relationship expressed in Equation 4-1 is graphically shown in Figure 4.30.

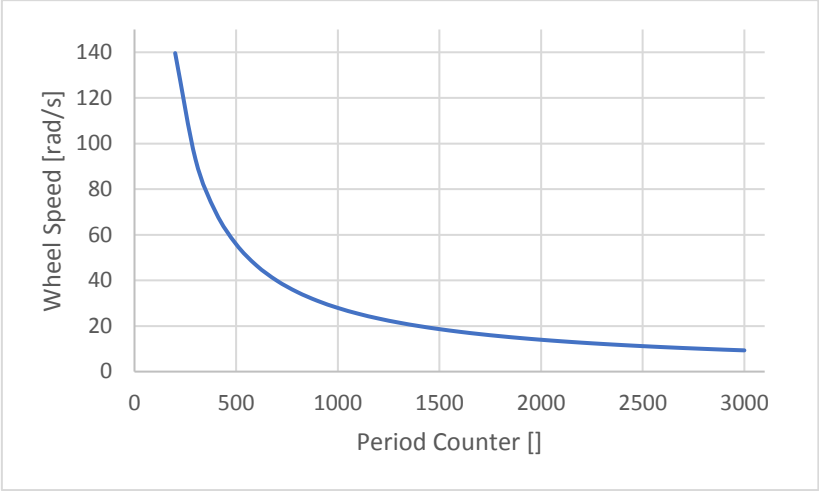


Figure 4.30 Period – Wheel Speed relationship

In Figure 4.31 the Real-Time function is shown. It gets the period update from FPGA every 10 ms, along with the validity check. Then, the signal is converted as described and filtered through a moving window algorithm and, finally, the values are stored in a global variable.

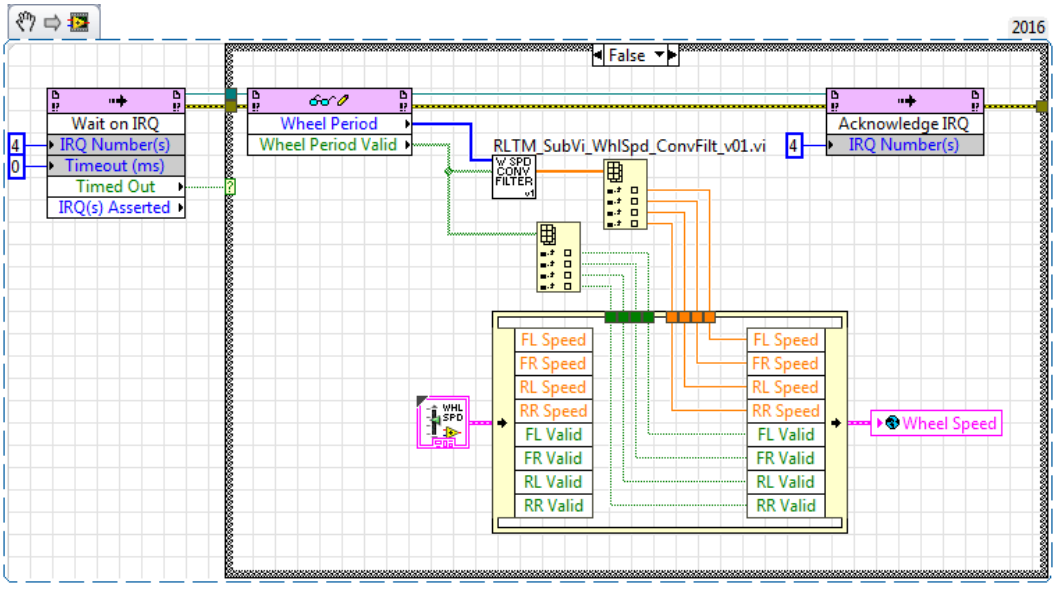


Figure 4.31 Real Time loop for wheel speed estimation

4.6.3 Low Level CAN-bus functions

As previously described, four CAN networks are connected to the UCP in order to exchange information with the different field devices. The low-level coding for the field bus management is here reported.

4.6.3.1 "CAN #1" Receive

As summarized in Table 4.1, the "CAN#1" connects the central controller with the two separate BMSs and the vehicle human-machine interface (MFU). Considering the receiving functions, Figure 4.32 shows the FPGA loop in charge of receiving frames from the CAN network "CAN#1".

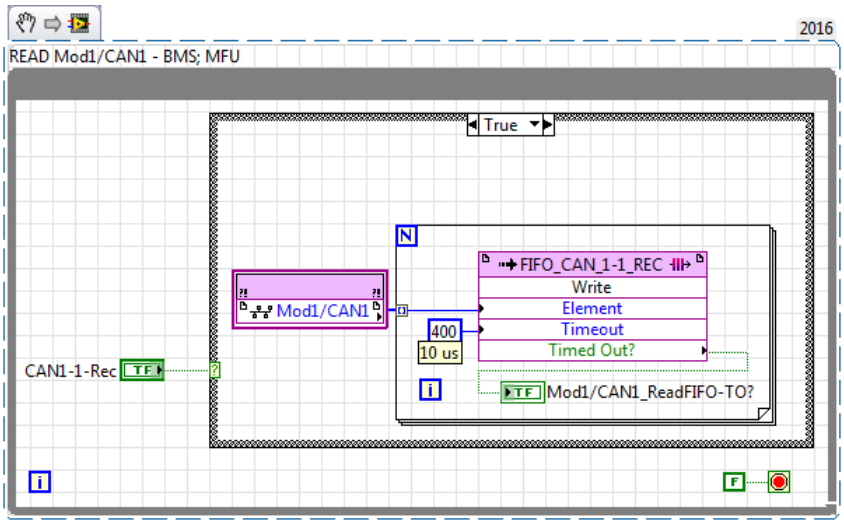


Figure 4.32 FPGA loop for the CAN #1 frame receiving

The frames are buffered in a FIFO as soon as they are received. This FIFO, in turn, is emptied by a RT loop, as seen in Figure 4.33.

In the real-time loop, the conversion of the data field is performed according to the frame arbitration identifier and the CAN specifications.

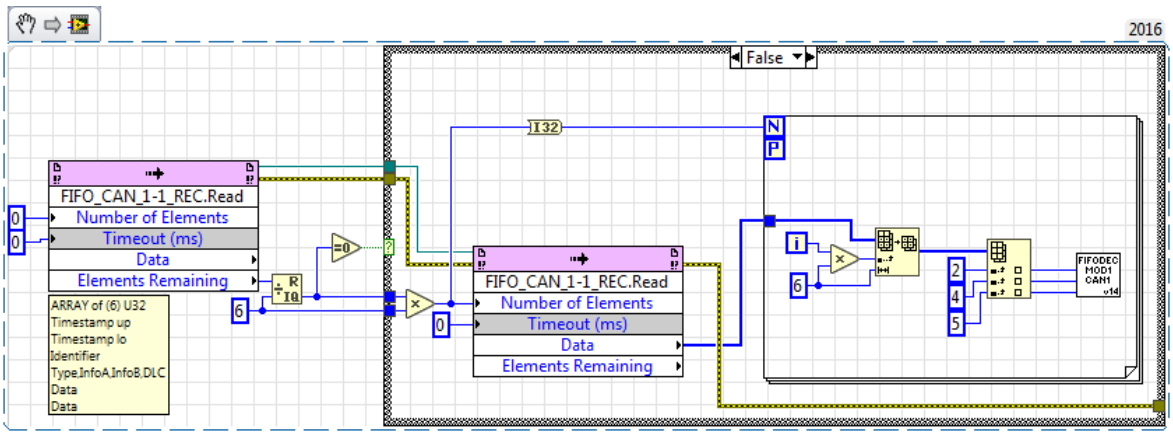


Figure 4.33 Real Time loop for the CAN #1 frame receiving

4.6.3.2 "CAN #3" Receive

CAN network "CAN#3" allows the information exchange between the UCP, the stability control actuators, the serial to CAN interface for the IMU sensor and a telemetry device.

To be more precise, the UCP receives only frames coming from the IMU interface, while it sends one reference frame to the TCSS and eleven frames to the telemetry system.

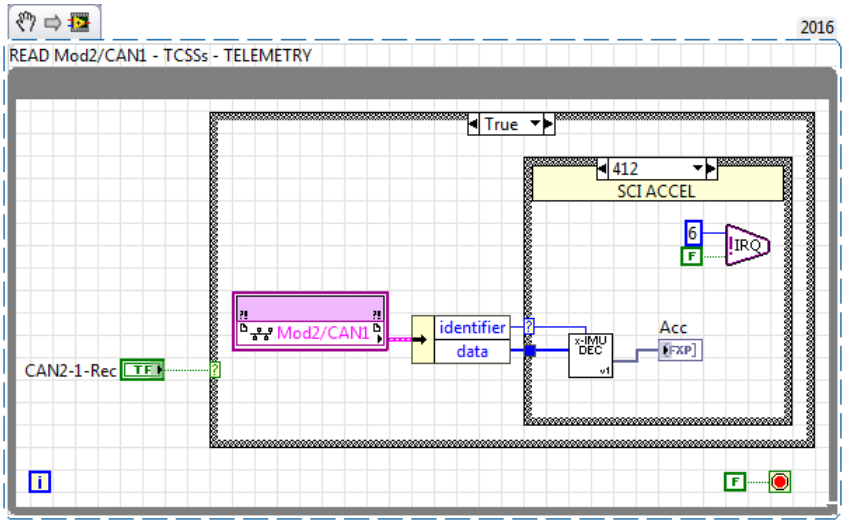


Figure 4.34 FPGA loop for the CAN #3 frame receiving

Figure 4.34 shows the receiving functions for the CAN frames broadcasted by the IMU CAN interface.

Unlike "CAN#1", where the frames are queued in a FIFO and processed by RT, the frames received on "CAN#3", containing the *x-IMU* outputs, are directly processed on the FPGA, so that no operation are required on the RT side. This is shown in Figure 4.35.

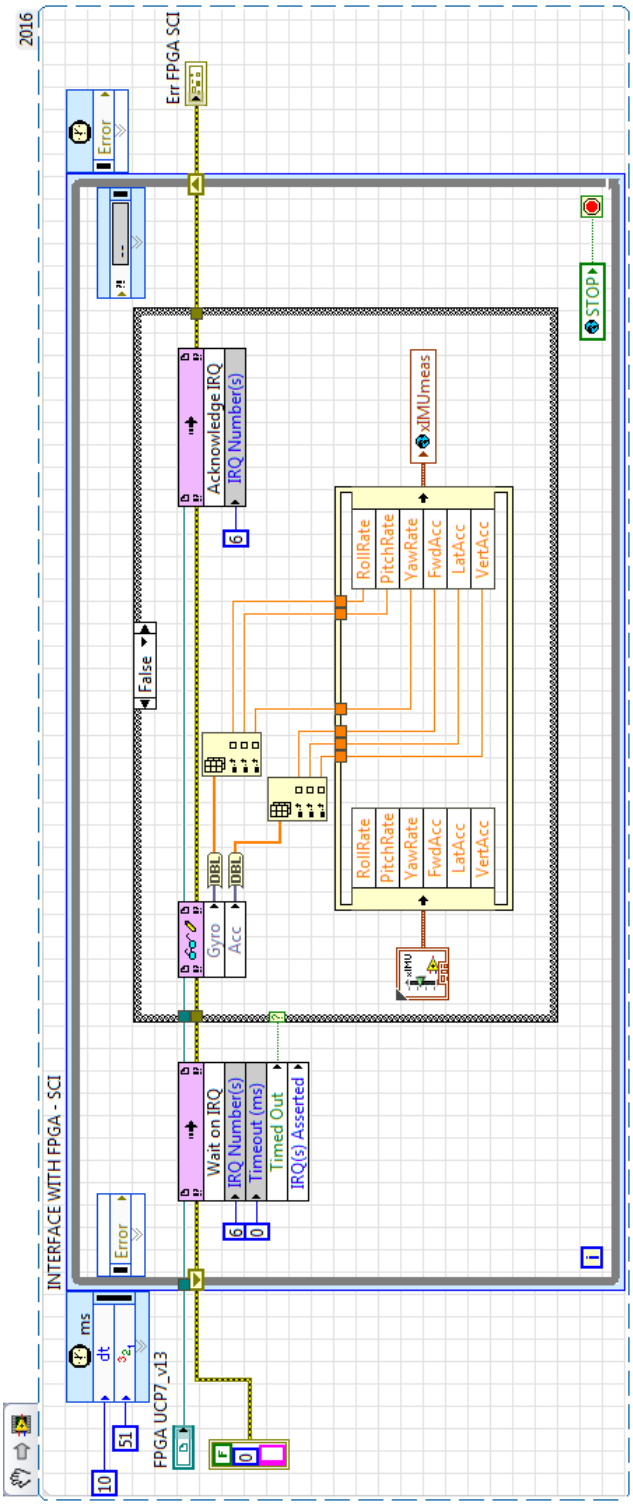


Figure 4.35 Real Time loop for the CAN #3 frame receiving

4.6.3.3 "CAN #4" Receive

On the AMBER-ULV system, "CAN#4" is reserved for communication between the two traction drives and the UCP. Traction drives send all the relevant states and measures, while UCP sends the torque reference command.

The receiving frame is managed in the same way as described for the "CAN#1". This means that the received frames are buffered in a FIFO which is read directly by the RT target and frame conversions are performed on that target. This logic is reported in Figure 4.36 and Figure 4.37.

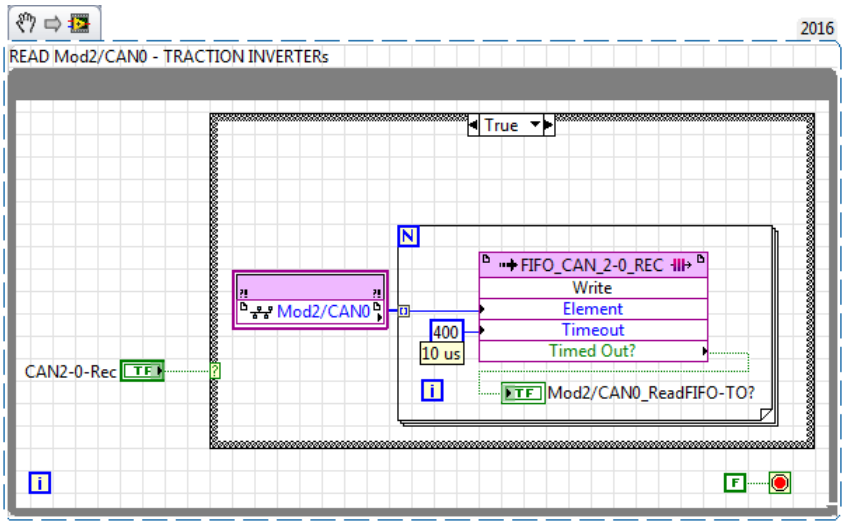


Figure 4.36 FPGA loop for the CAN #4 frame receiving

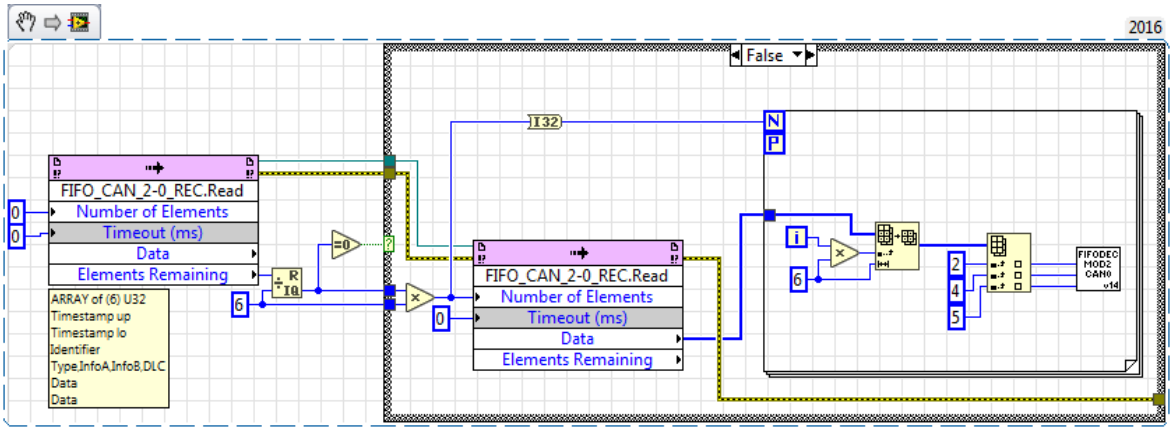


Figure 4.37 Real Time loop for the CAN #4 frame receiving

4.6.3.4 "CAN #5" Receive

The Electric Power Steering receives from UCP the enable command and the vehicle speed information (for adjusting the response) and returns the actual steering angle.

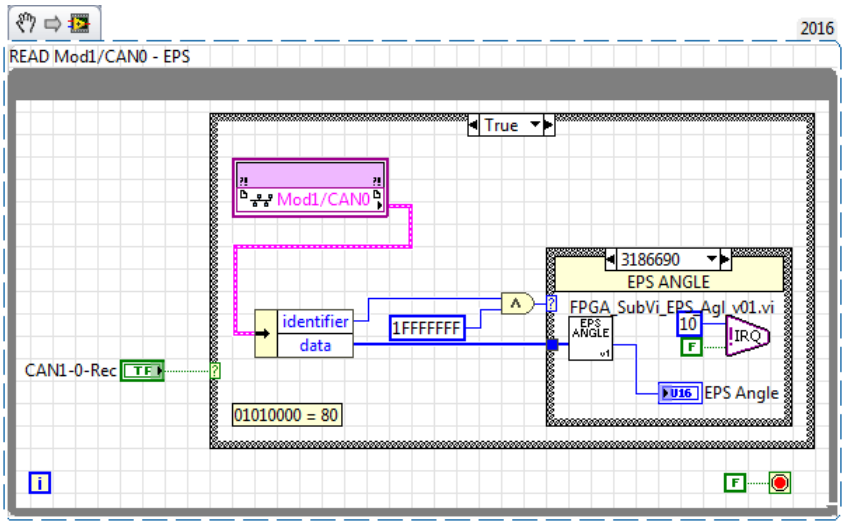


Figure 4.38 FPGA loop for the CAN #5 frame receiving

The frame containing the steering angle is directly converted on FPGA (Figure 4.38) and the RT loop is updated with an interrupt.

The same RT loop that receives information from EPS, also sends to FPGA the enable command and the vehicle speed (Figure 4.39).

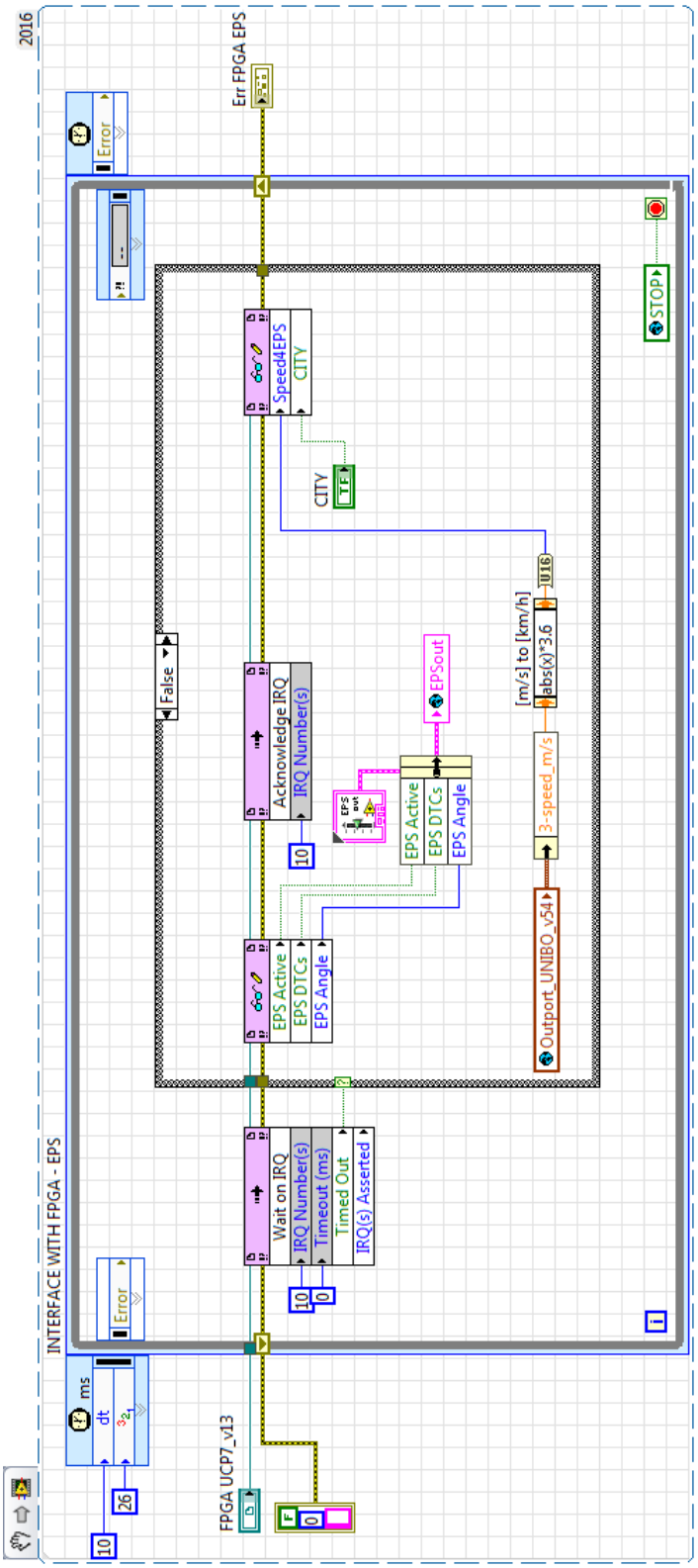


Figure 4.39 RT loop for the CAN #5 frame receiving

4.6.3.5 "CAN#1" Transmission

The UCP sends through CAN-Bus "CAN#1" vehicle information which are used by the MFU for vehicle control and Human-Machine Interface purposes.

The variables to be transmitted to MFU are scaled and converted in an integer representation within the RT loop shown in Figure 4.40.

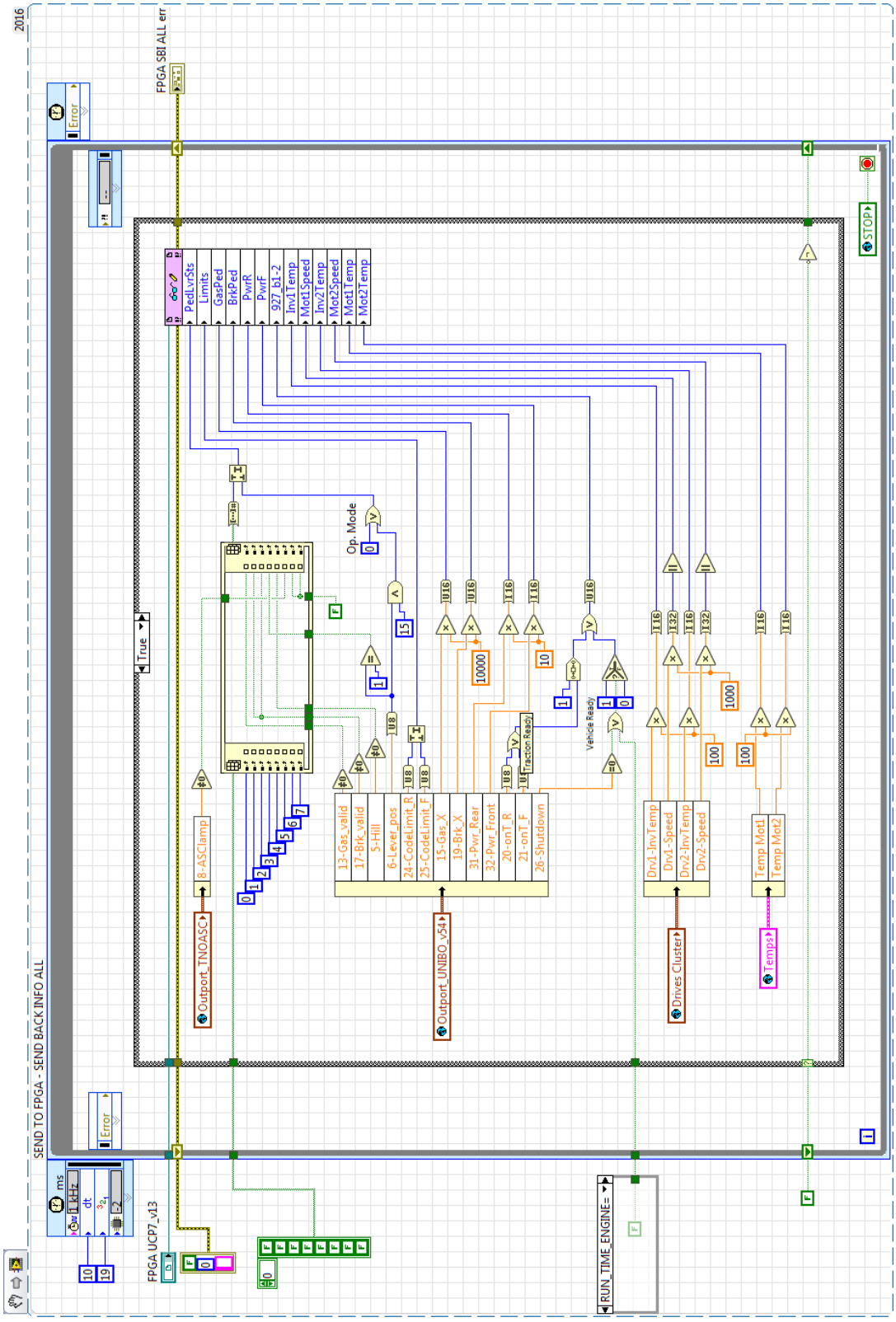


Figure 4.40 Real Time loop for the CAN #1 frame transmitting

On the FPGA side, a FPGA loop is implemented for each frame to be transmitted to the UCP (Figure 4.41). Each loop is in charge of properly filling the data field of each defined frame and transmit it to the CAN network.

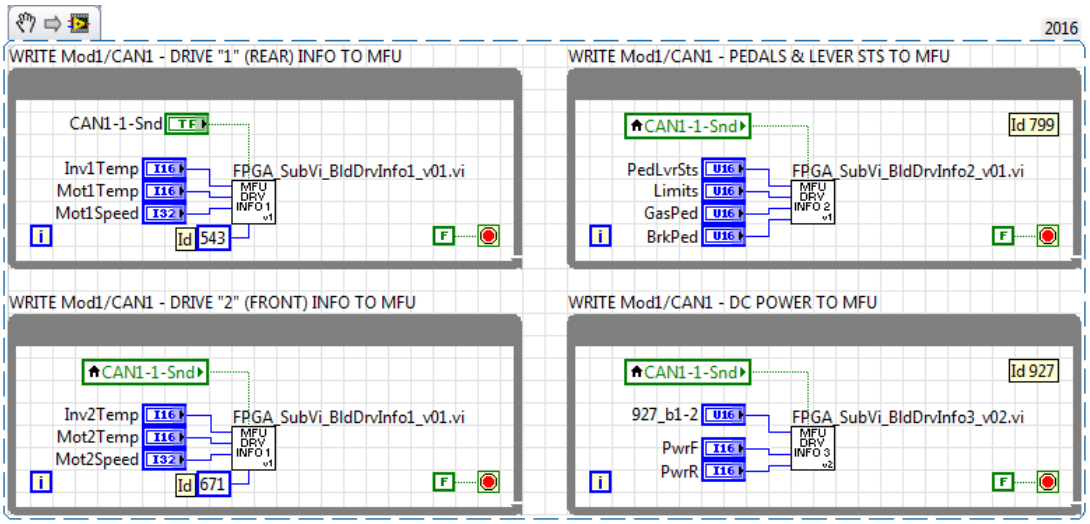


Figure 4.41 FPGA loop for the CAN #1 frame transmitting

4.6.3.6 "CAN#3" Transmission

As mentioned earlier, the UCP sends braking torque references through "CAN#3" to the TCSS actuator.

The functions for this task are shown in the following Figure 4.42 to Figure 4.48.

For the "CAN#3" bus, the RT side is developed with the same strategy presented for "CAN#1". The RT function of Figure 4.42 shows the single conversion applied to the TCSS references, that is the double precision float to integer conversion.

Figure 4.43 and Figure 4.44 show the RT loops scaling and updating to FPGA the variables for the Telemetry system. Also, some information from the MFU loop (Figure 4.40) are bridged over "CAN#3".

Figure 4.45 to Figure 4.48, moreover, show the FPGA loops for the transmission of the TCSS reference and Telemetry frame.

"CAN#3" is the network with the highest number of frames sent by UCP. Most of them also have the same transmission rate of 100 Hz. For this reason, each frame, composed through dedicated function on FPGA, is buffered into a FIFO memory. The CAN transmitting function takes the frames to be sent from that FIFO.

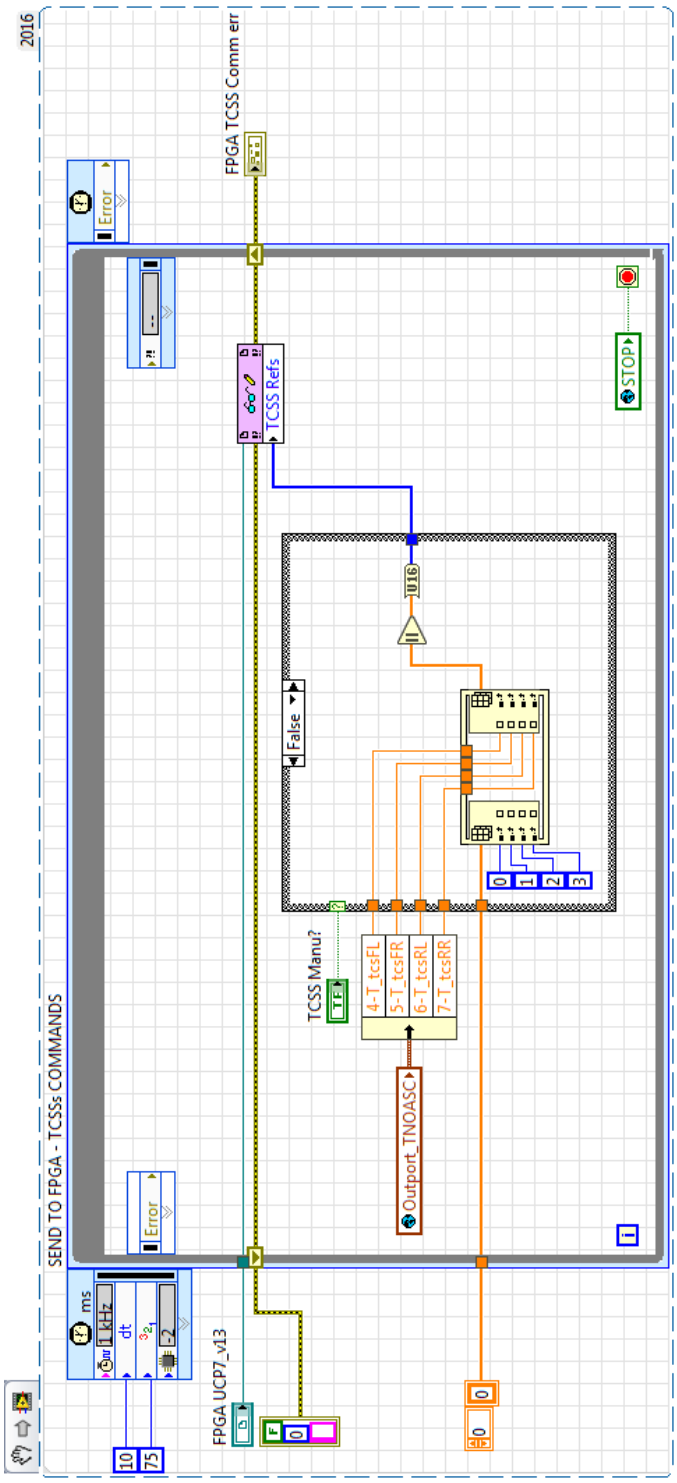


Figure 4.42 Real Time loop for the CAN #3 frame transmitting (TCSS reference)

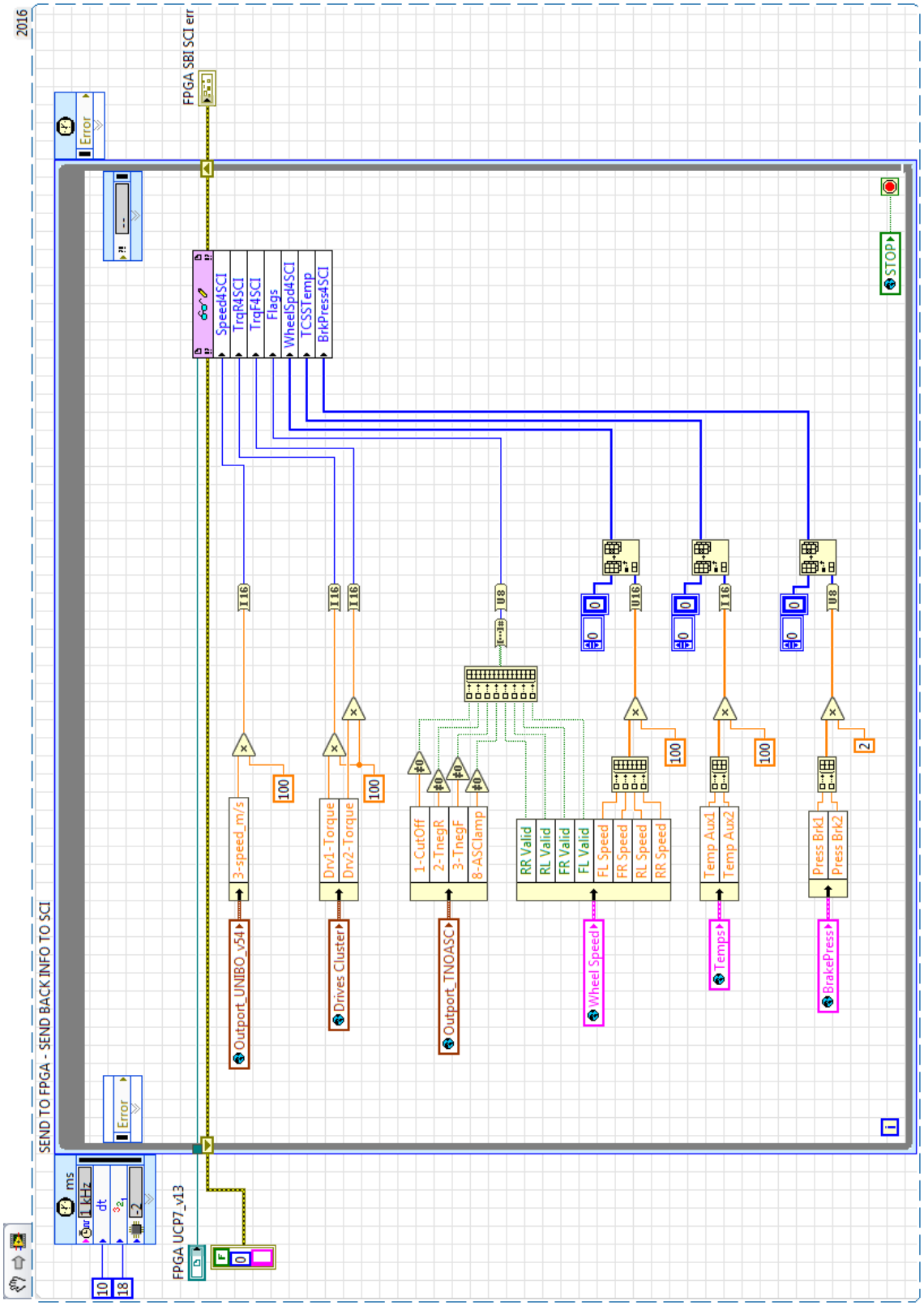


Figure 4.43 Real Time loop for the CAN #3 frame transmitting (Telemetry-1)

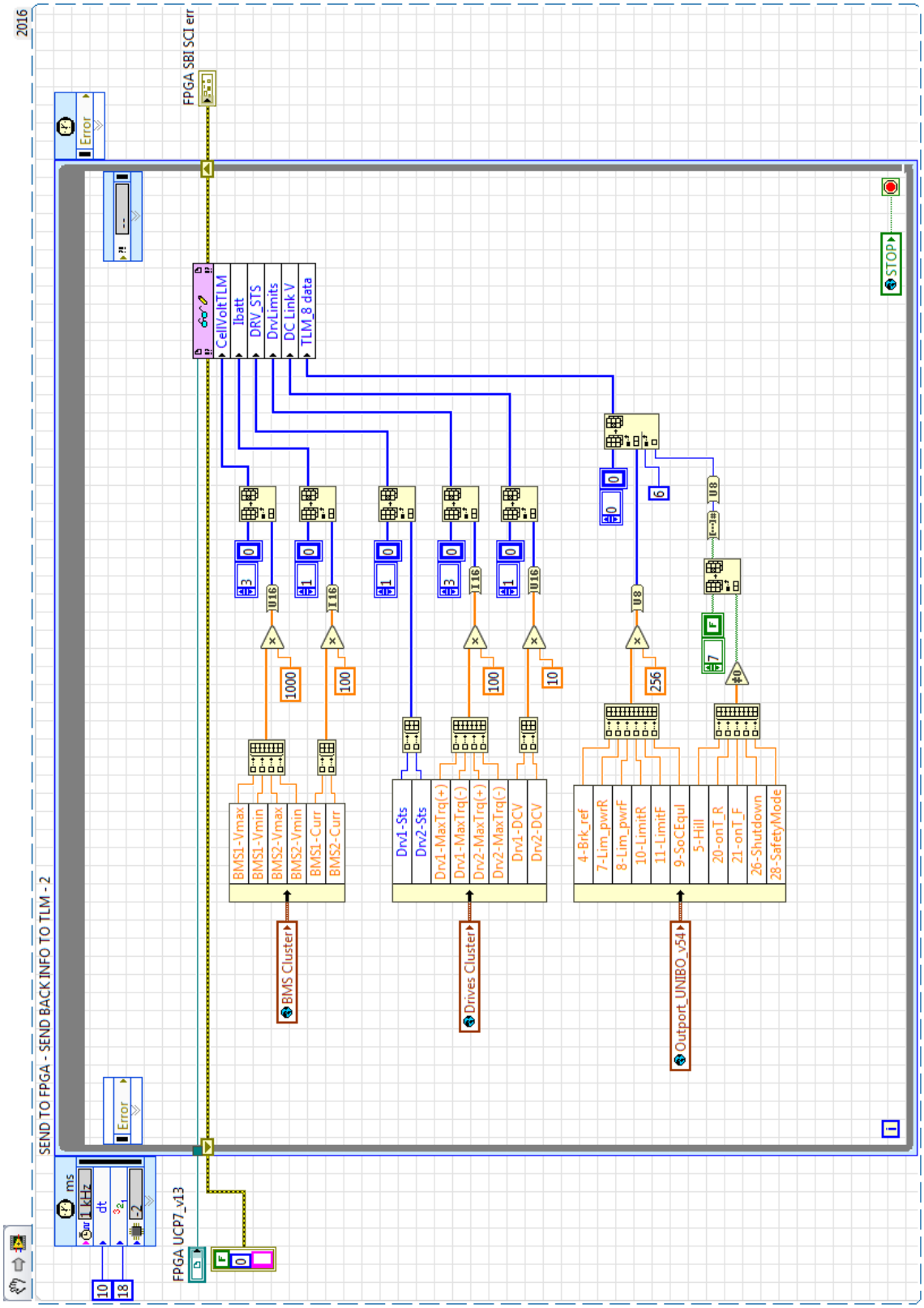


Figure 4.44 Real Time loop for the CAN #3 frame transmitting (Telemetry-2)

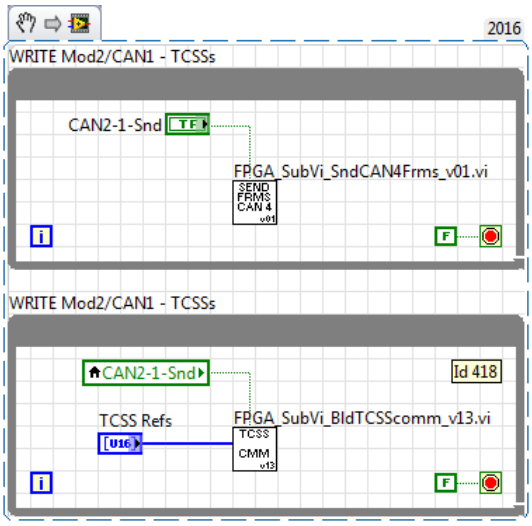


Figure 4.45 FPGA loop for the CAN #3 frame transmitting (1)

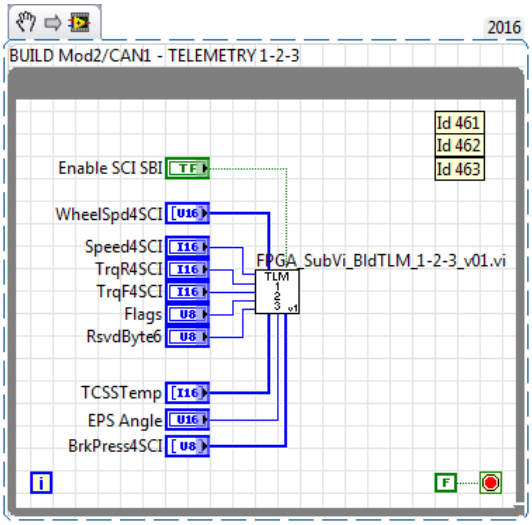


Figure 4.46 FPGA loop for the CAN #3 frame transmitting (2)

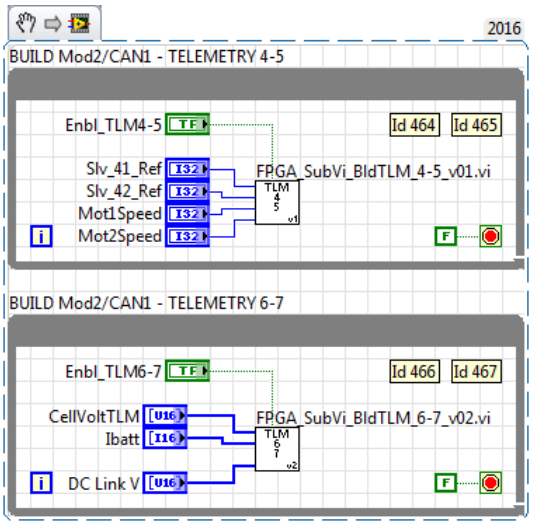


Figure 4.47 FPGA loop for the CAN #3 frame transmitting (3)

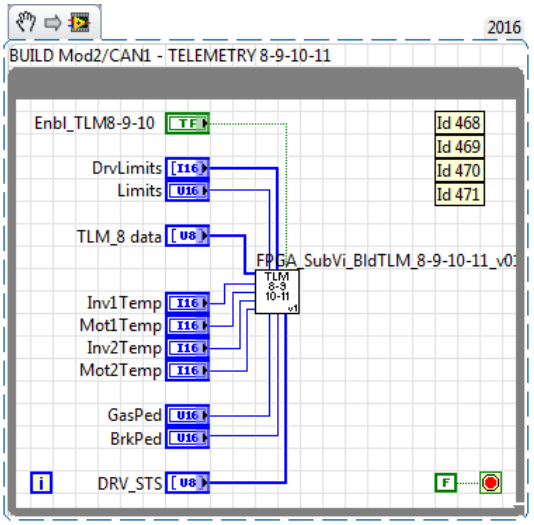


Figure 4.48 FPGA loop for the CAN #3 frame transmitting (4)

4.6.3.7 "CAN#4" Transmission

For each traction drive, the commands sent to them are the *power on request*, which determine whether or not the traction drive is enabled to follow a torque reference, and the torque reference itself. The logic is shown in Figure 4.49 and Figure 4.50.

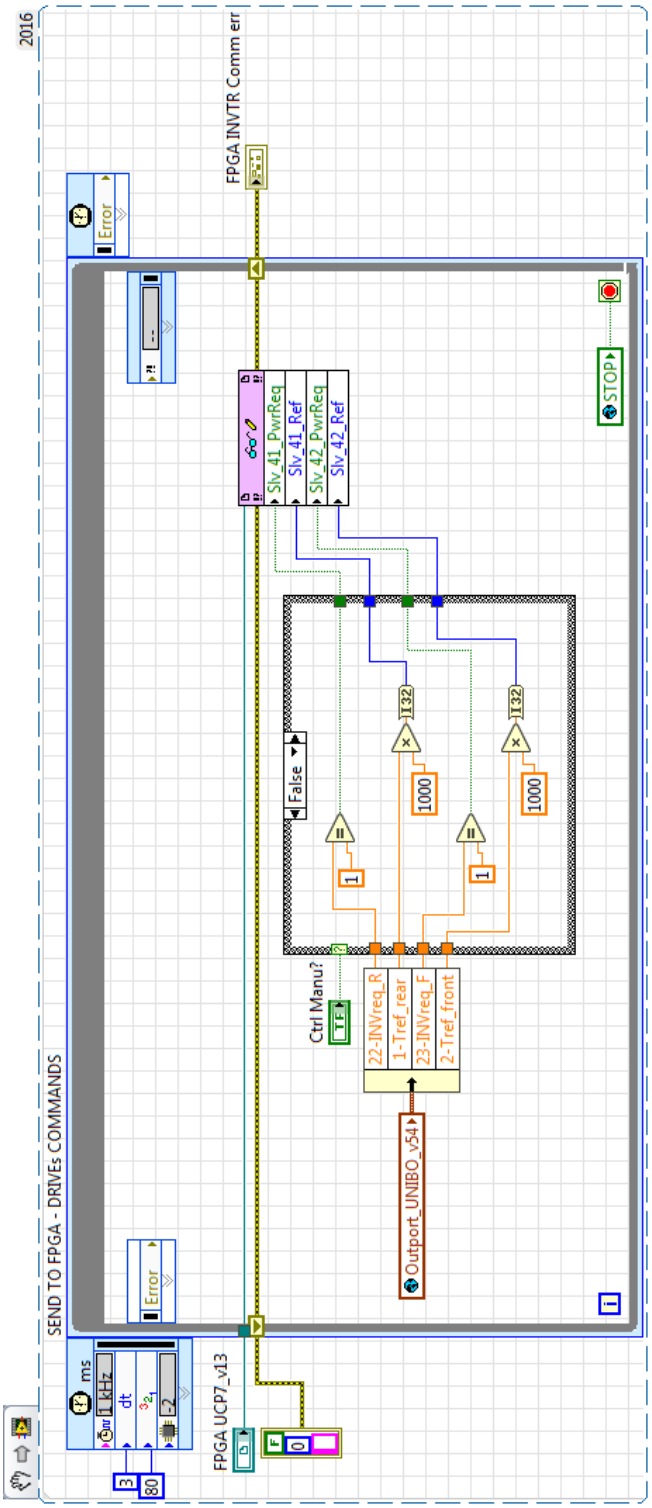


Figure 4.49 RT loop for the CAN #4 frame transmitting

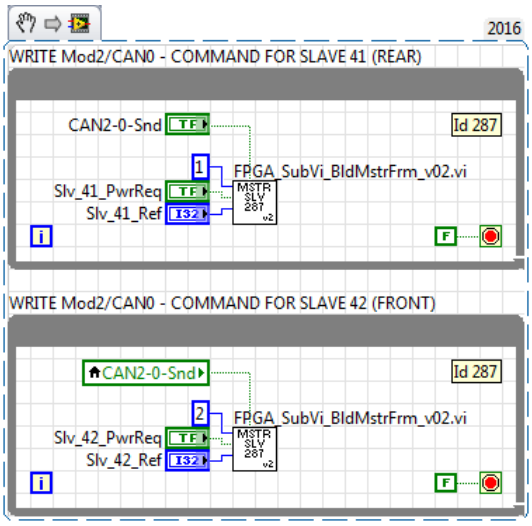


Figure 4.50 FPGA loop for the CAN #4 frame transmitting

4.6.3.8 "CAN#5" Transmission

The EPS enable frame and vehicle speed frame are built and sent, on the FPGA side, as displayed in Figure 4.51. This information comes from the RT thanks to the same receiving loop of Figure 4.39.

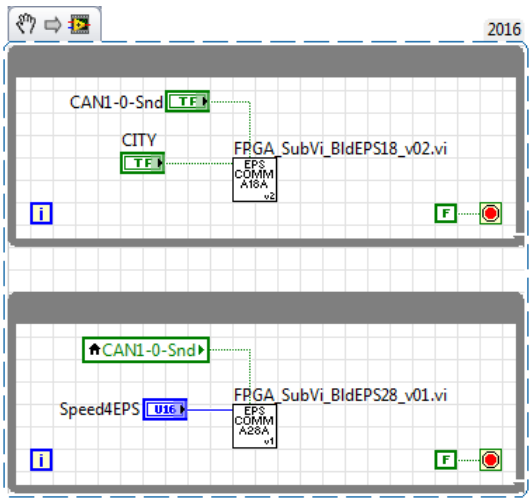


Figure 4.51 FPGA loop for the CAN #5 frame transmitting

4.6.4 High-level functions interface

As described in Chapter 3, the *integration* process also included the development of a low-level interface for the automatically generated high-level code.

Figure 4.52 shows the low-level interface for both high-level algorithms, namely the *Traction Management* (described in Chapter 2) and the stability control (provided by the partner).

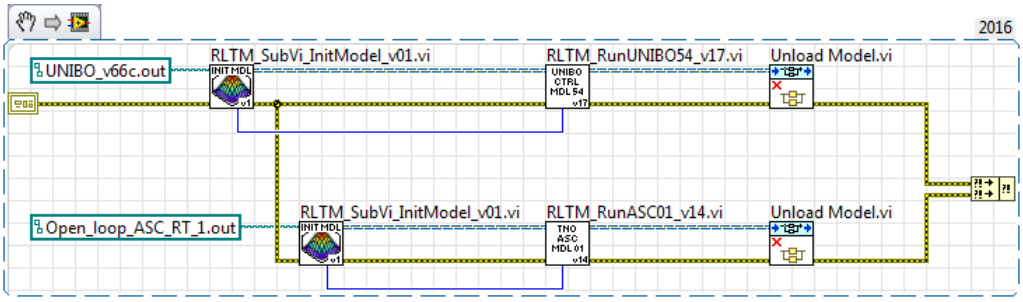


Figure 4.52 Interface for high-level algorithms

For both the high-level algorithms, a start up function is implemented which can be seen on the left side of Figure 4.52 and enlarged in Figure 4.53. This loads the model library from the path it is stored in the controller's memory, opens the "parameter interface" allowing on-line tuning of model parameters and, critically, gets the model period (set in the pre-build configuration), which will be used as loop timing, ensuring the correct behaviour of the compiled model.

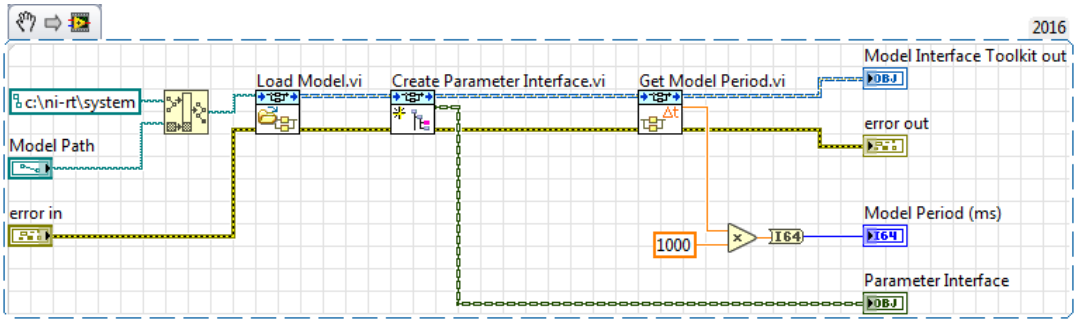


Figure 4.53 High-level code initialization

4.6.4.1 Traction Management

The Traction Management algorithm is, therefore, run as shown in Figure 4.54. As explained in Chapter 3, the LabVIEW function only treats the input ports and output ports as floating point array. For ensuring the correct address of each variable, a dedicated function was developed, shown in Figure 4.55.

The output array is indexed in a dedicated global variable to make the model output variables available for the whole control algorithm.

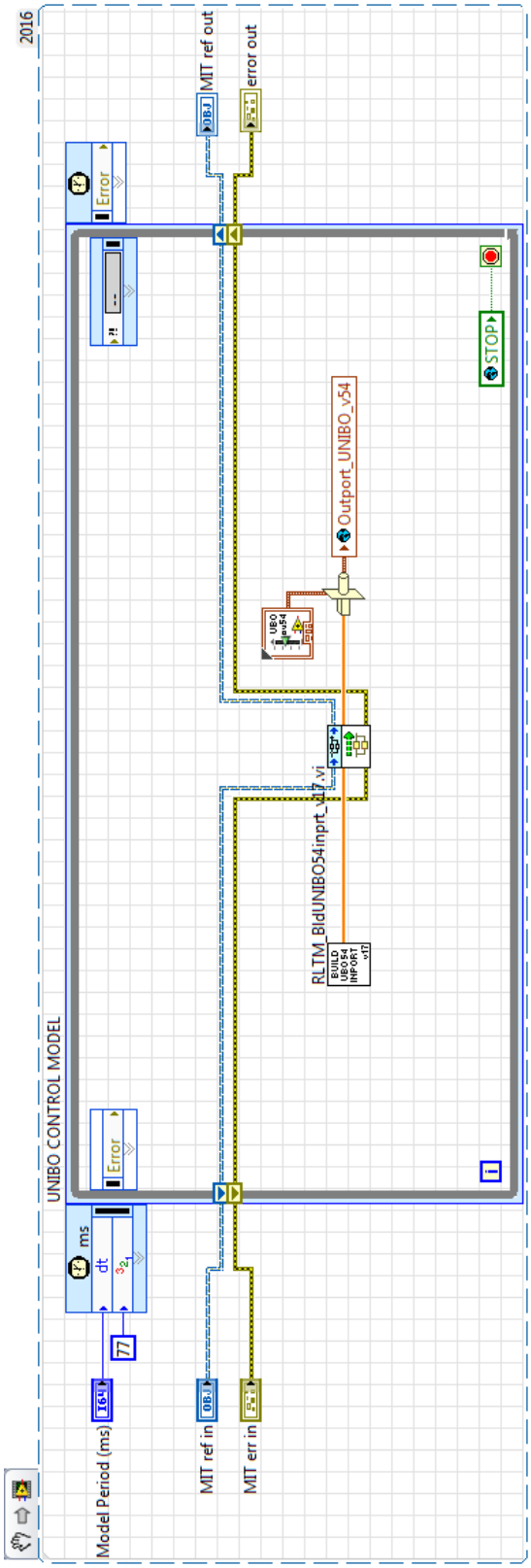


Figure 4.54 Traction Management algorithm execution

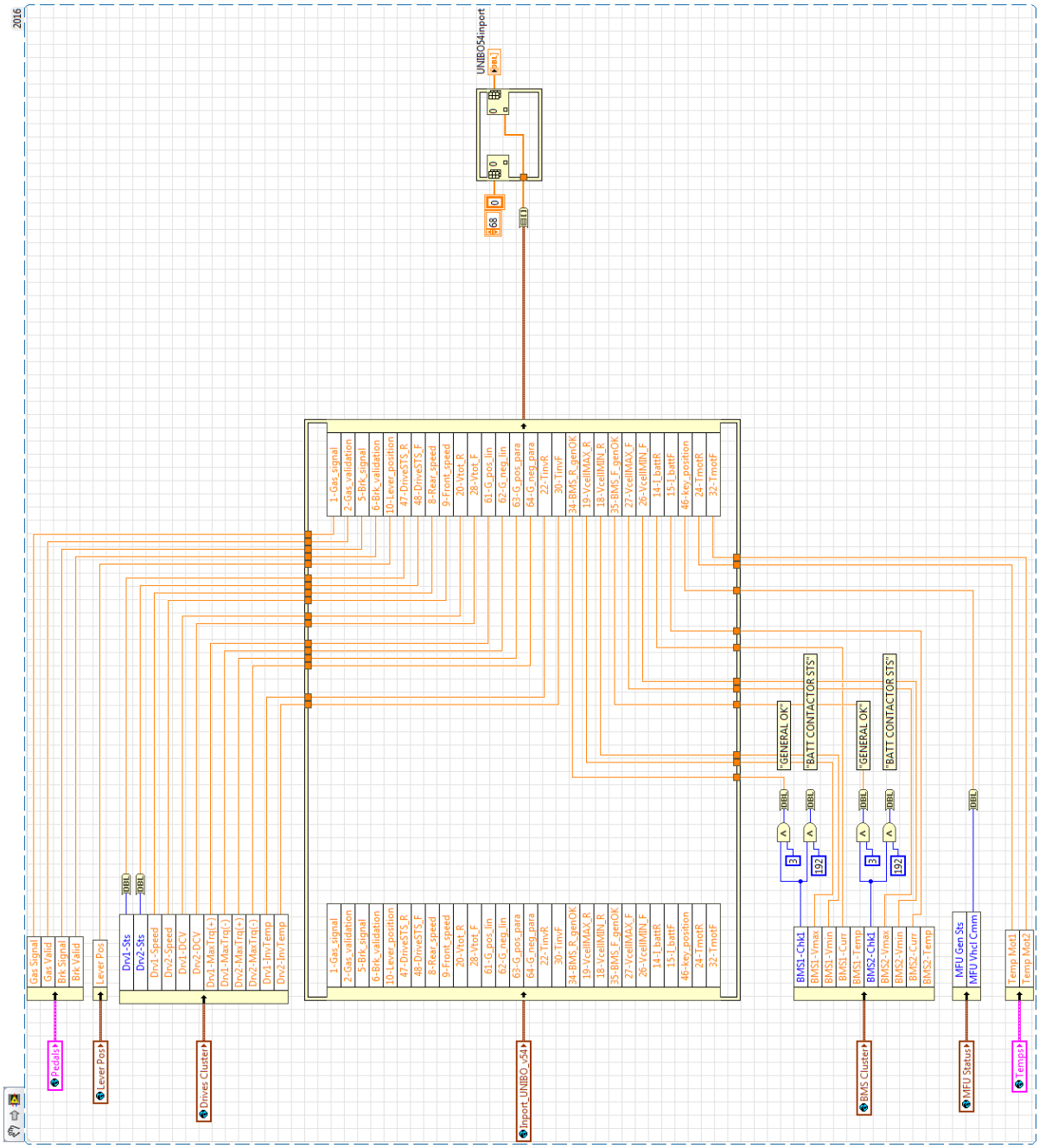


Figure 4.55 Traction Management input array update

4.6.4.2 Stability Control

The Stability Control function is handled at low-level exactly as the Traction Management is, so that Figure 4.56 shows how the built model is executed, while Figure 4.57 displays the input array indexing with all the required variables.

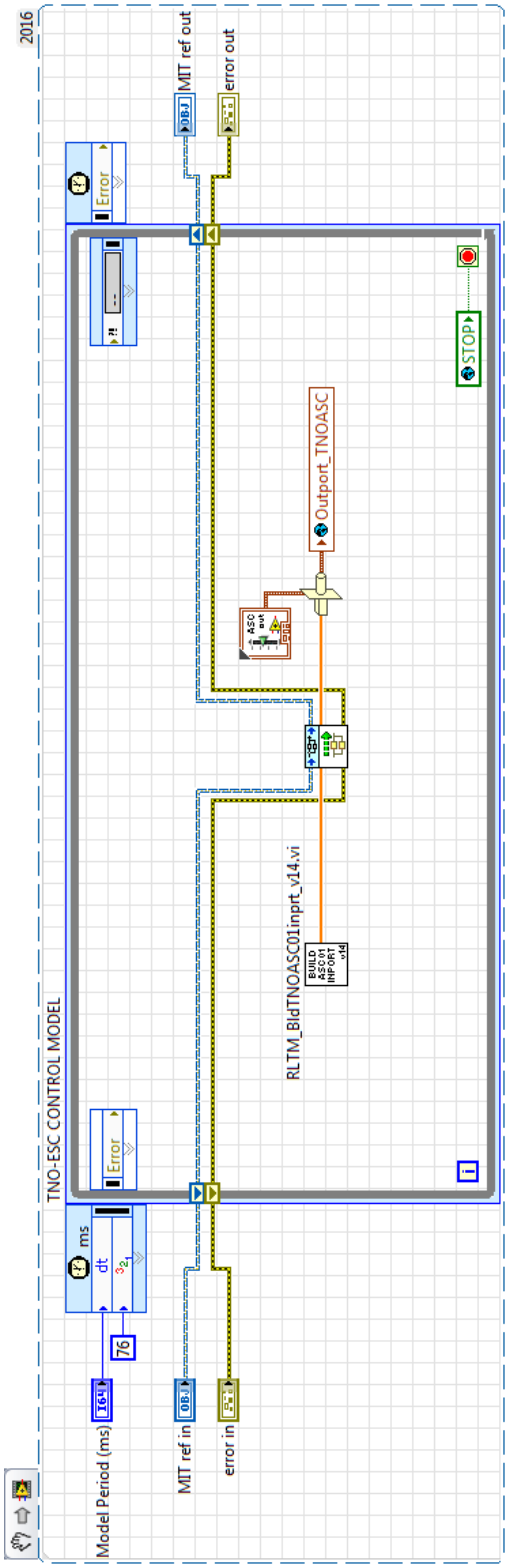


Figure 4.56 Stability Control algorithm execution

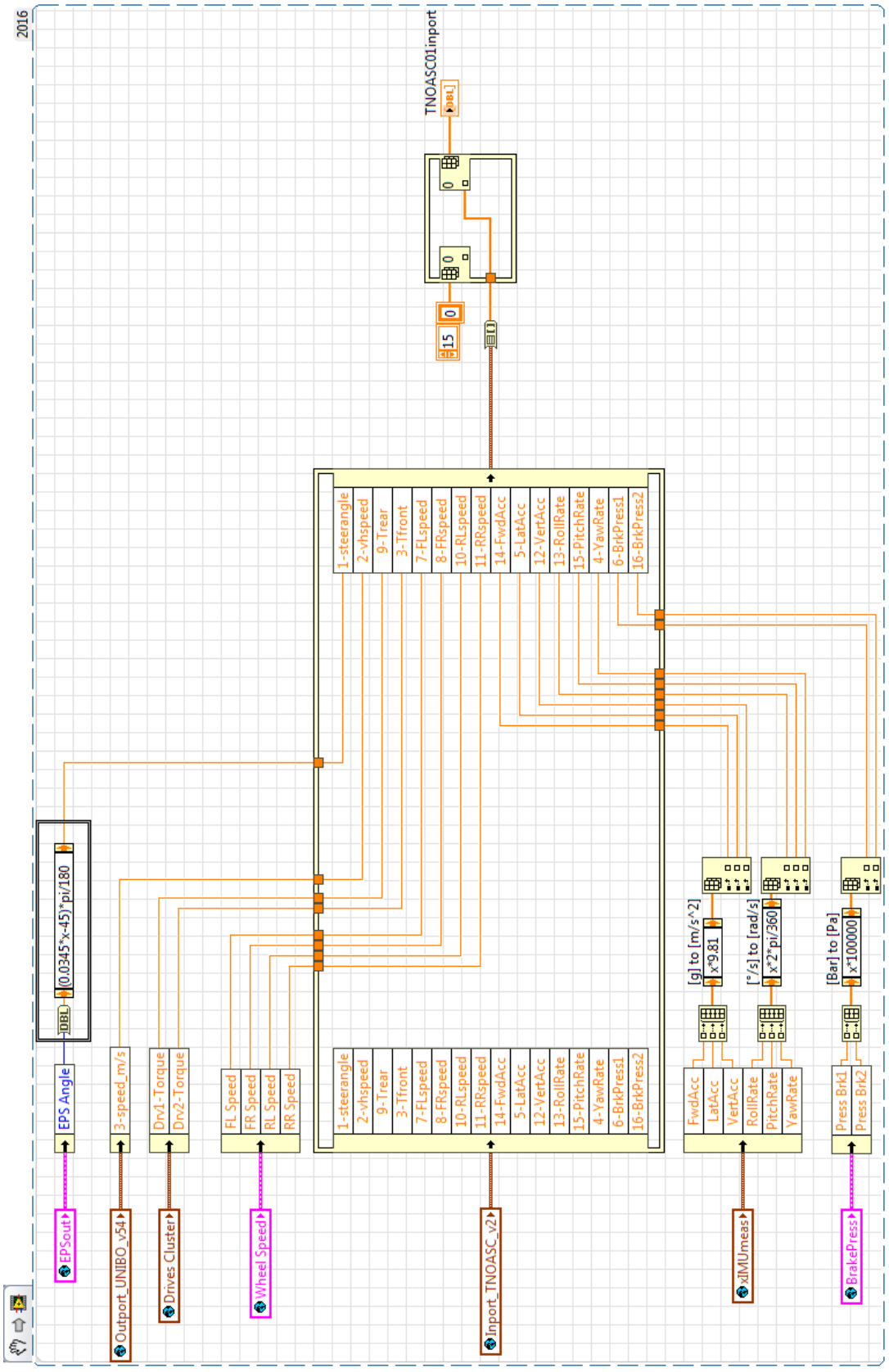


Figure 4.57 Stability Control input array update

Chapter 5
Prototype demonstration tests

The research activities performed during the Ph.D. and described in the previous chapters, were applied for the deployment on a demonstration prototype, part of the AMBER-ULV project.

The electric vehicle was extensively tested, aiming to prove the successful outcomes of the development.

Among the performed mechanical, industrial, safety and electric tests, the followings will be described in this section, relevant to this dissertation:

- Component tests;
- Vehicle “shakedown” tests;
- Acceleration;
- Energy consumption characteristics;
- Real traffic performance;
- Stability.

Figure 5.1 AMBER-ULV “T3” demonstration prototype

5.1 Component test

As described in Chapter 4, all the sensors, hardware connections and field buses were tested at the LEMAD laboratory before their installation on the vehicle prototype.

A component-level check was carried out after the prototype final assembly as well, proving the correct installation of the low-level devices.

All the voltage levels, scaling and conversions performed by the high-level traction management software were checked and recorded. The communication between the controller ECU and the CAN bus devices was tested both in "reading" and "writing" mode, the latter correctly performed the transmission of manual commands to the various devices.

Figure 5.2 shows the controller box, also containing the hardware connection circuitry described in Chapter 4.

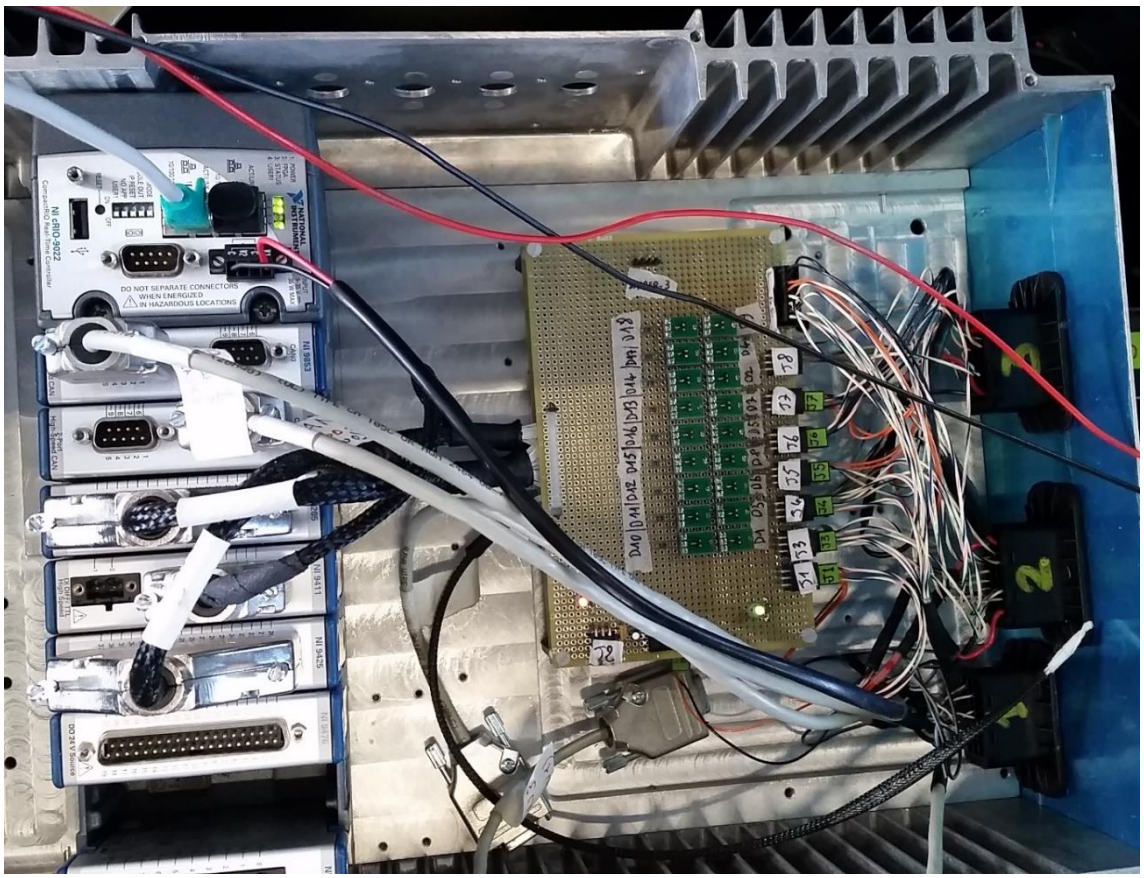


Figure 5.2 Example of sensor's hardware circuitry on the vehicle demonstrator

5.2 Telemetry system

For the security during tests and for post-processing purposes, a telemetry and data-log system was set up on the AMBER-ULV prototype. This was accomplished by the vehicle ECU broadcasting over a CAN network some of its internal variables and most of the measurements. The broadcasted quantities are listed in Table 5.1 and Table 5.2

A PC interface able to display these quantities (Figure 5.3) and log them in a spreadsheet file was developed for the purpose.

Index	Variable	Meas. Unit	Notes
1	Timestamp	[s]	
2	Vehicle speed	[m/s]	
3	Rear requested torque	[Nm]	
4	Front requested torque	[Nm]	
5	Rear actual torque	[Nm]	
6	Front actual torque	[Nm]	
7	Rear maximum positive torque	[Nm]	
8	Rear maximum negative torque	[Nm]	
9	Front maximum positive torque	[Nm]	
10	Front maximum negative torque	[Nm]	
11	Rear drive speed	[rpm]	
12	Front drive speed	[rpm]	
13	Rear battery current	[A]	
14	Front battery current	[A]	
15	Rear battery voltage	[V]	
16	Front battery voltage	[V]	
17	Rear maximum cell voltage	[V]	
18	Rear minimum cell voltage	[V]	
19	Front maximum cell voltage	[V]	
20	Front minimum cell voltage	[V]	
21	Gas pedal position	[pu]	
22	Brake pedal position	[pu]	
23	"*brk_ref*"	[pu]	Traction Management internal variable
24	"*brk_ref*"	[pu]	Traction Management internal variable
25	"*Lim_pwr R*"	[pu]	Traction Management internal variable
26	"*Lim_pwr F*"	[pu]	Traction Management internal variable
27	"*Limit R*"	[pu]	Traction Management internal variable
28	"*Limit F*"	[pu]	Traction Management internal variable
29	"*SoC equl*"	[pu]	Traction Management internal variable
30	"*rear drive status*"	[code]	Traction Management internal variable
31	"*front drive status*"	[code]	Traction Management internal variable
32	"*Rear limitation*"	[code]	Traction Management internal variable
33	"*Front limitation*"	[code]	Traction Management internal variable
34	"*Hill*"	[flag]	Traction Management internal variable
35	"*on T R*"	[flag]	Traction Management internal variable
36	"*on T F*"	[flag]	Traction Management internal variable
37	"*Shutdown*"	[flag]	Traction Management internal variable
38	"*Safety mode*"	[flag]	Traction Management internal variable
39	"*Spare1*"	[flag]	Traction Management internal variable
40	"*Spare2*"	[flag]	Traction Management internal variable
41	"*Spare3*"	[flag]	Traction Management internal variable

Table 5.1 Telemetry and logged variables

Index	Variable	Meas. Unit	Notes
42	Rear inverter temperature	[°C]	
43	Rear motor temperature	[°C]	
44	Front inverter temperature	[°C]	
45	Front motor temperature	[°C]	
46	Rear TCSS temperature	[°C]	
47	Front TCSS temperature	[°C]	
48	Brake pressure circuit 1	[bar]	
49	Brake pressure circuit 2	[bar]	
50	Steering angle	[code]	
51	TCSS LF torque request	[Nm]	Stability Control Output
52	TCSS RF torque request	[Nm]	Stability Control Output
53	TCSS LR torque request	[Nm]	Stability Control Output
54	TCSS RR torque request	[Nm]	Stability Control Output
55	TCSS LF status	[code]	
56	TCSS RF status	[code]	
57	TCSS LR status	[code]	
58	TCSS RR status	[code]	
59	TCSS LF actual value	[deg]	
60	TCSS RF actual value	[deg]	
61	TCSS LR actual value	[deg]	
62	TCSS RR actual value	[deg]	

Table 5.2 Telemetry and logged variables

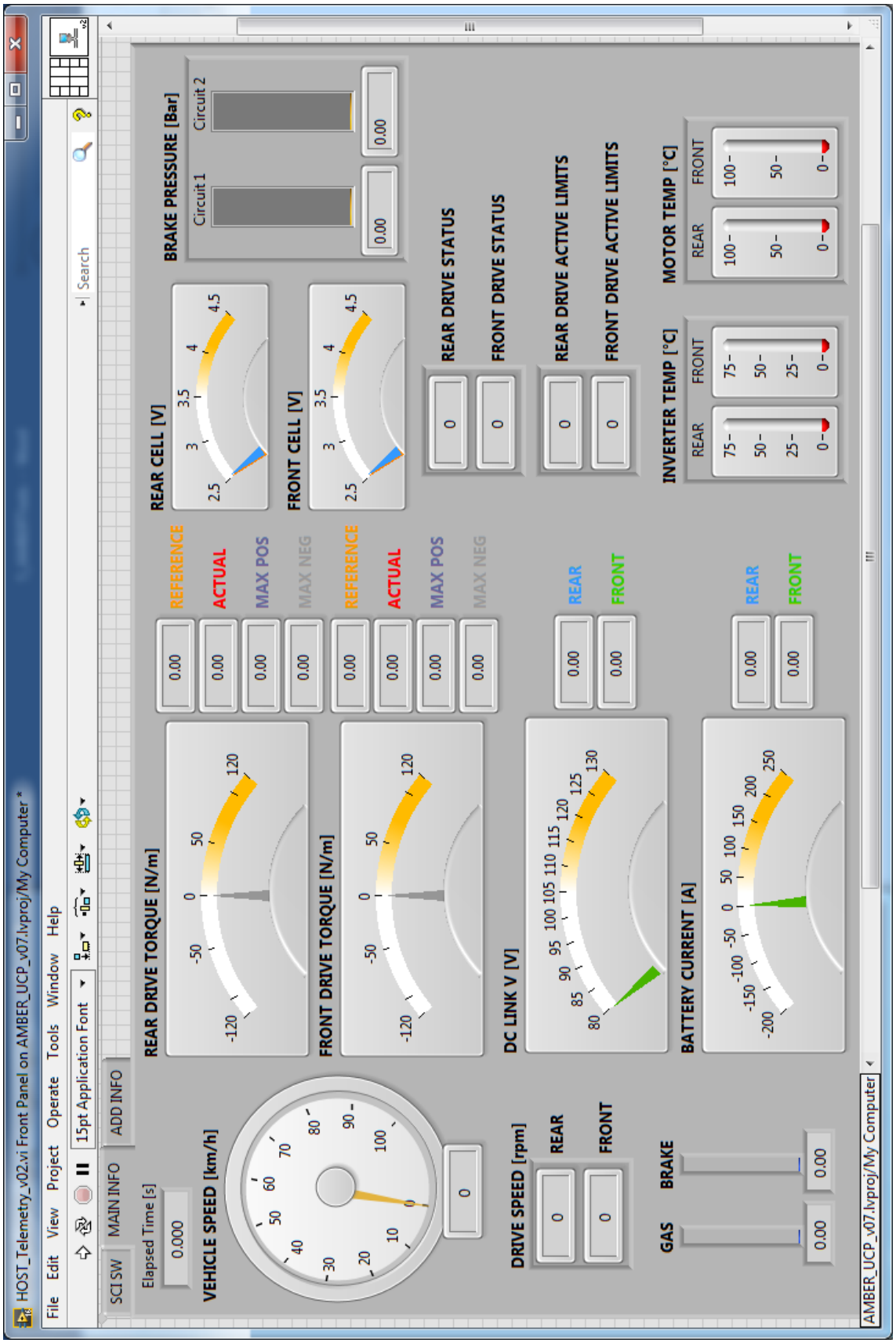


Figure 5.3 Telemetry interface

5.3 Shakedown

The *shakedown* represents the first real-environment test of a prototype. Its objective is to demonstrate that the system can operate all-in-all and to bring to light as many design-induced early failures as possible.

For the AMBER-ULV prototype, the shakedown test was conducted for several days, both on closed circuit performing predefined speed profiles and on public roads, under real traffic conditions.

During these tests, the telemetry was continuously monitored, in order to detect any abnormalities as soon as possible, while the logged data were post-processed between test sessions. Hereinafter, some results are proposed.

As it can be seen from Figure 5.4, the presented test is constituted by four accelerations and decelerations. According to the Gas and Brake pedal position (see Figure 5.6), the two drives deliver positive or negative torque, so that energy is recovered during decelerations.

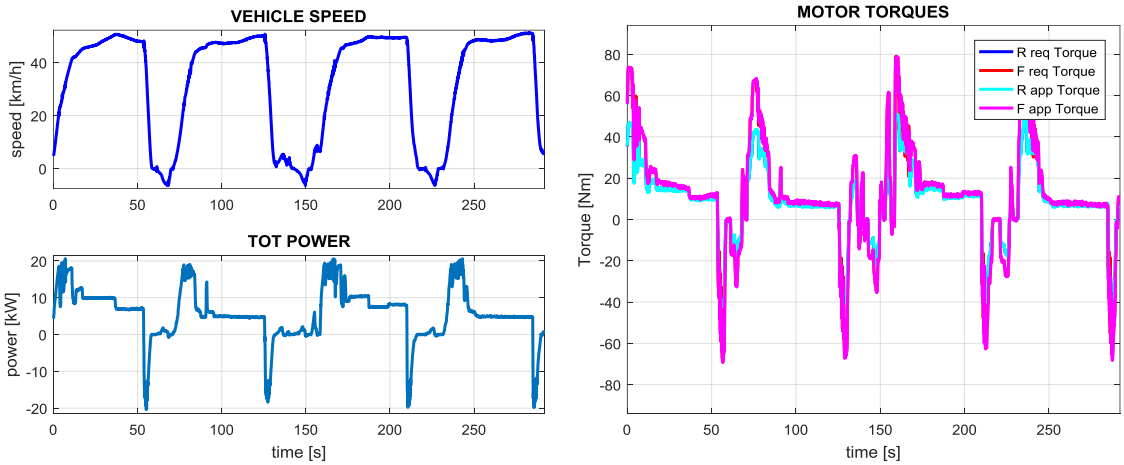


Figure 5.4 Shakedown: Speed, power, motor torques

Figure 5.5 displays the individual battery power exchanged by both battery packs (left part of the graphs), where the energy recovery is highlighted, although blended with the fiction brakes during very sudden decelerations, as evinced from the brake pressure graph.

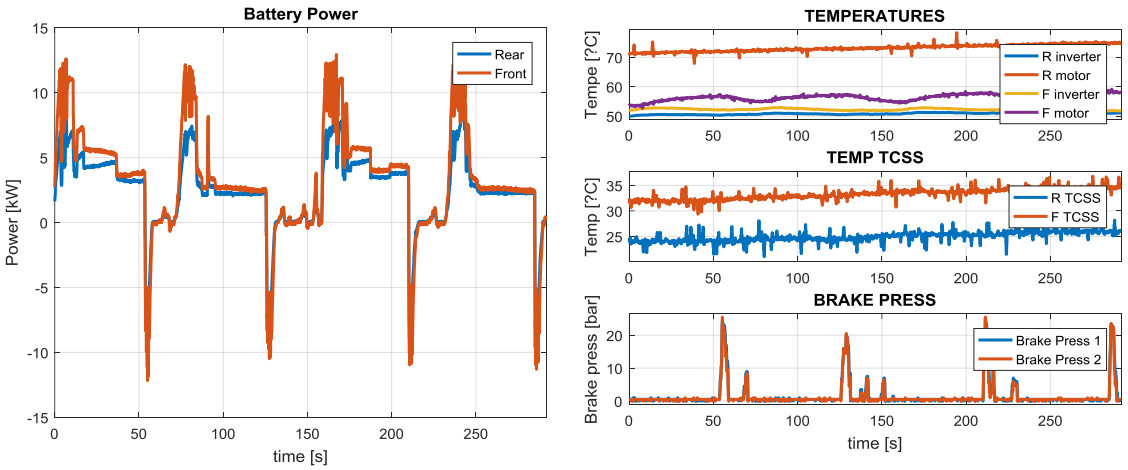


Figure 5.5 Shakedown: Battery power, acquired temperatures, brake hydraulic pressure

Figure 5.5 also reveals an abnormality on the rear motor temperature (upper-right graph). During subsequent investigation, it turned out to be due to insufficient air flow to the rear motor assembly.

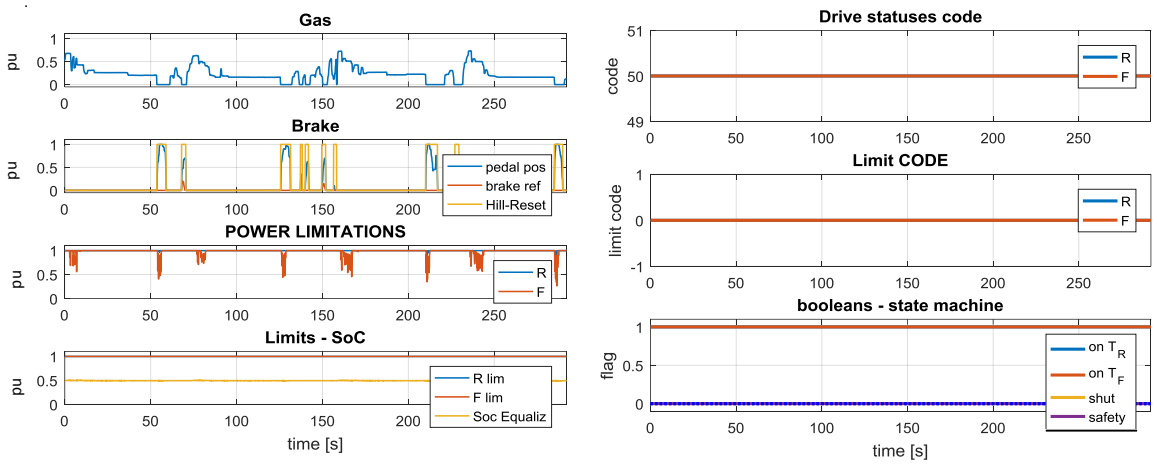


Figure 5.6 Shakedown: Pedal positions, traction management internal variables

5.4 Acceleration

The acceleration is one of the parameters to determine the performance of a vehicle. It consists in applying, with the vehicle at standstill, the maximum available force and measuring the time it takes to reach a certain speed or to travel a certain distance.

Several acceleration tests were carried out with the AMBER-ULV prototype. Figure 5.7 provides an example in which the variables of interest concerning the acceleration tests, recorded from the telemetry system, are displayed.

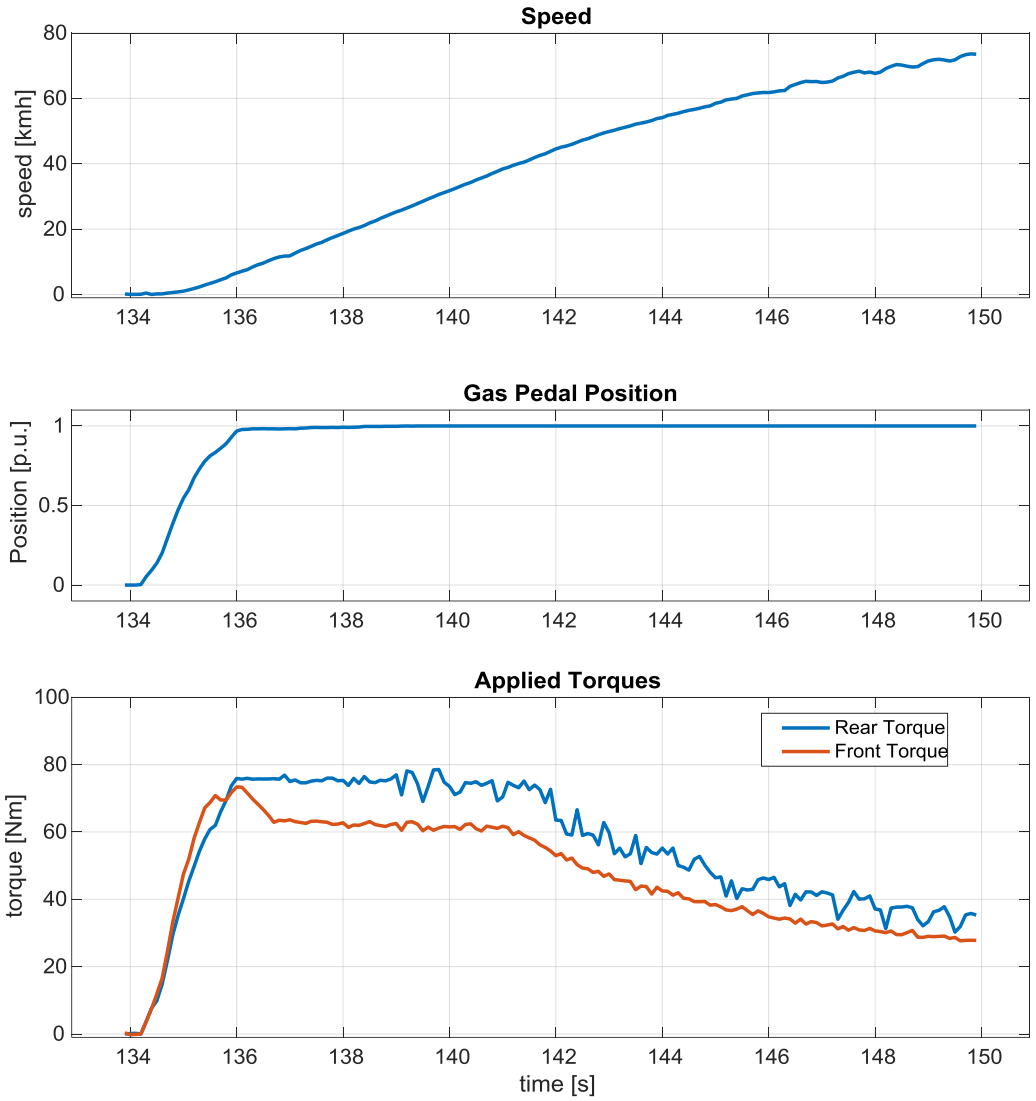


Figure 5.7 Acceleration: Vehicle speed, gas pedal position, motor torques

It should be noted that some parts of the prototype chassis were realized with metallic materials rather than composites as originally drafted. This led to an increase in weight compared to the initial design and, therefore, the acceleration performance was slightly lower than expected.

5.5 Energy consumption

The energy consumption is one of the most important features of any vehicle. The characteristic can be experimentally determined driving at constant speed on a flat and straight road and measuring the power used by the car.

By conducting this test at various speeds, the power vs. speed curve can be obtained.

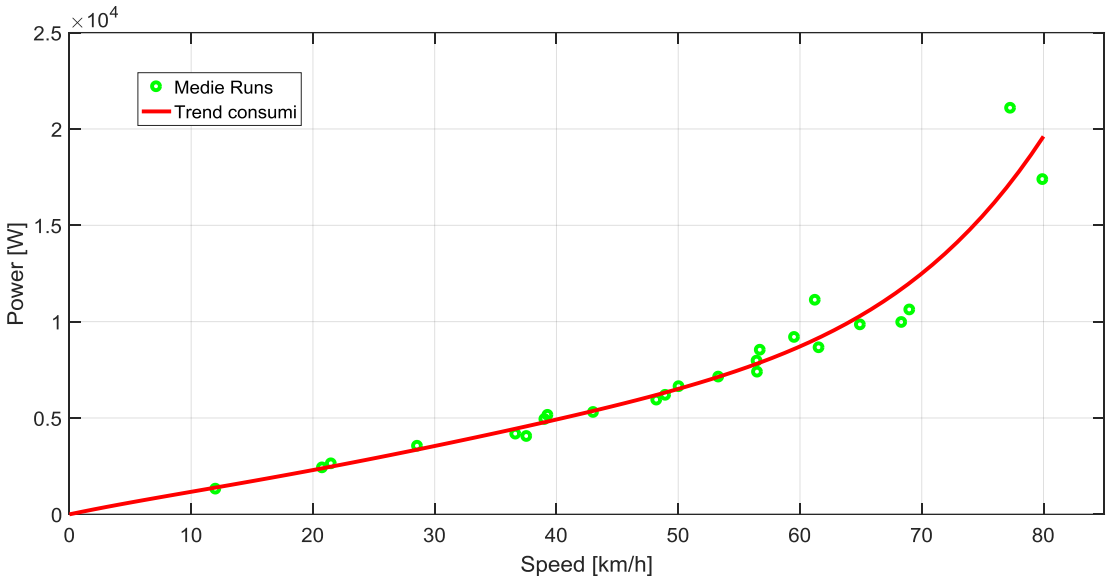


Figure 5.8 Prototype energy consumption characteristic

For the AMBER-ULV prototype, consumption tests were carried out on a suitable stretch road and produced the characteristic of Figure 5.8, obtained as a curve fitting of experimental power measures taken at different speeds up to 80 km/h.

Figure 5.9 Location of the experimental energy consumption measurements

5.6 Real traffic shakedown

After adequate tests on closed tracks, real-traffic tests were also performed on the public roads of Rome.

The following graphs are an excerpt obtained from the telemetry log.

Figure 5.10 displays the vehicle speed profile and the corresponding motor speed for both drives, while Figure 5.11 represent the requested and actual motor torques.

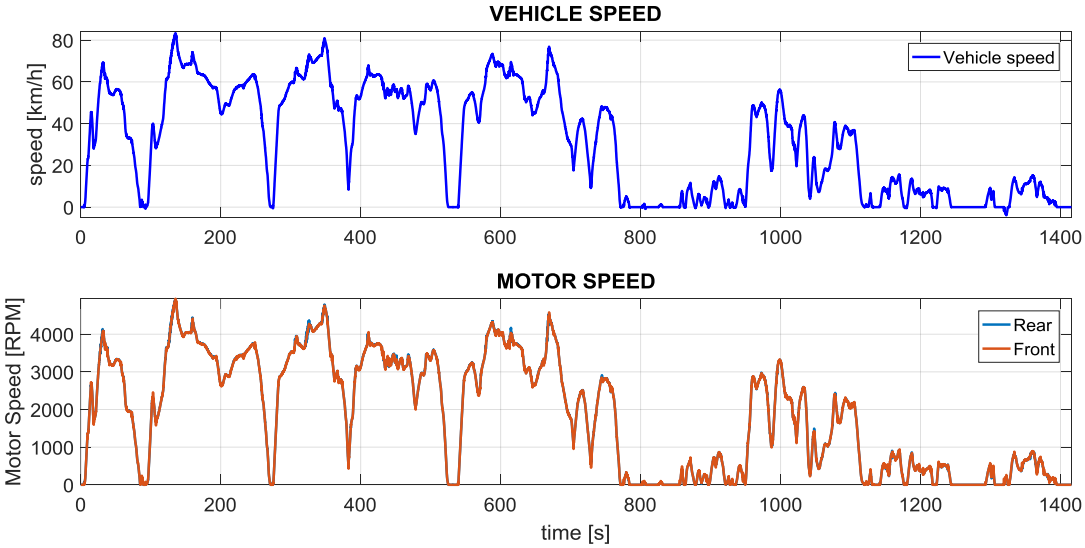


Figure 5.10 Road tests: Vehicle speed, motor speeds

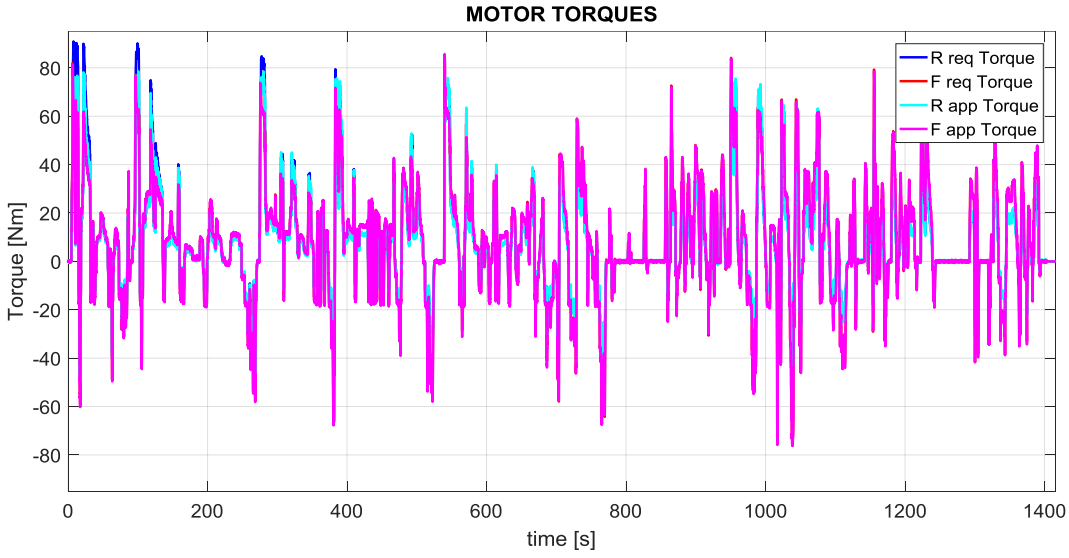


Figure 5.11 Road tests: motor torques

Figure 5.12 shows the instantaneous battery power (for both front and rear battery) obtained in post-processing as product of the battery current and the battery voltage, which are shown in Figure 5.13.

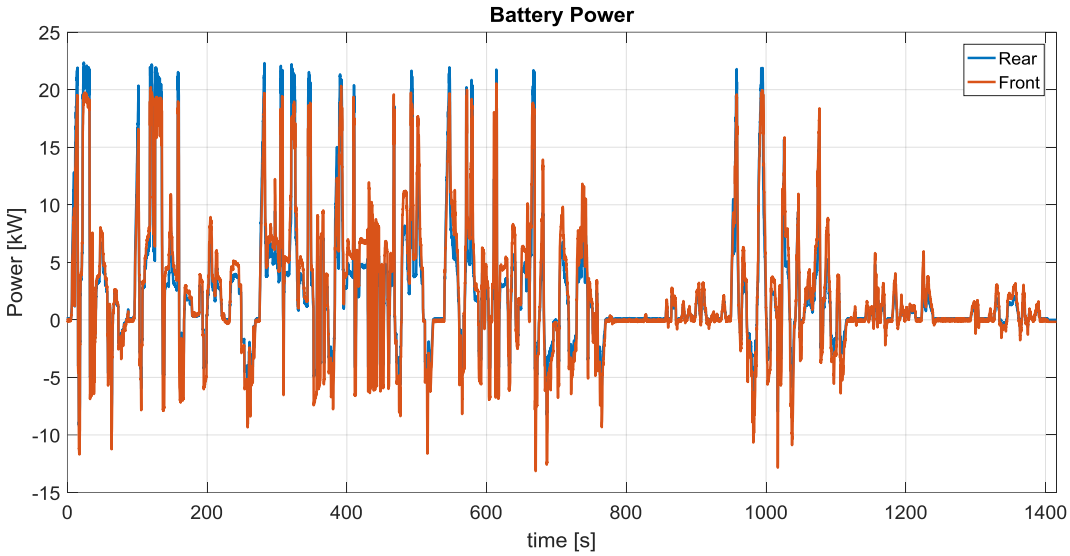


Figure 5.12 Road tests: Battery power

Figure 5.13 also shows the minimum and maximum cell voltages for the rear and the front battery.

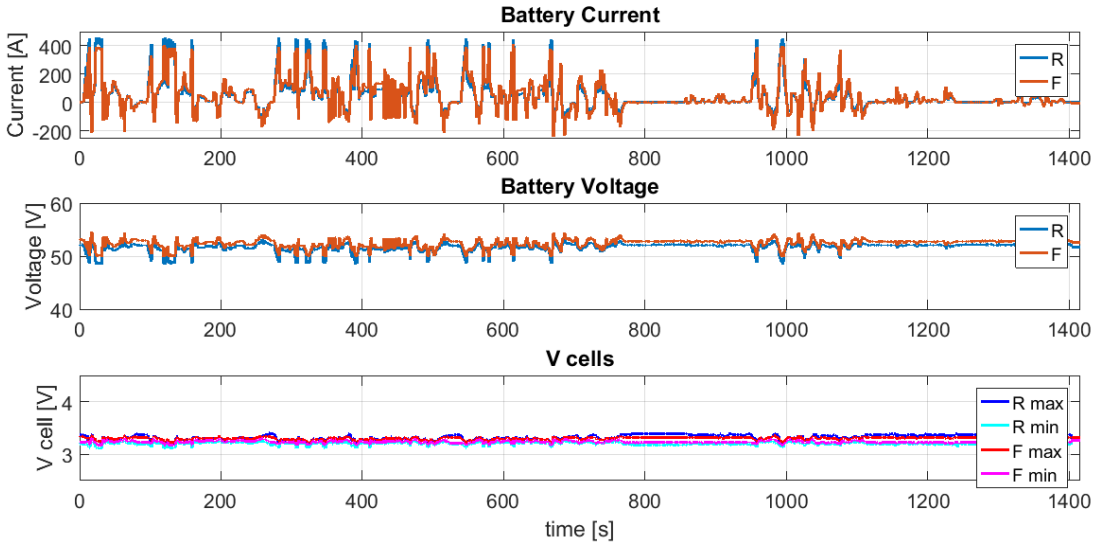


Figure 5.13 Road tests: Battery measurements (current, voltage, minimum and maximum cell voltages)

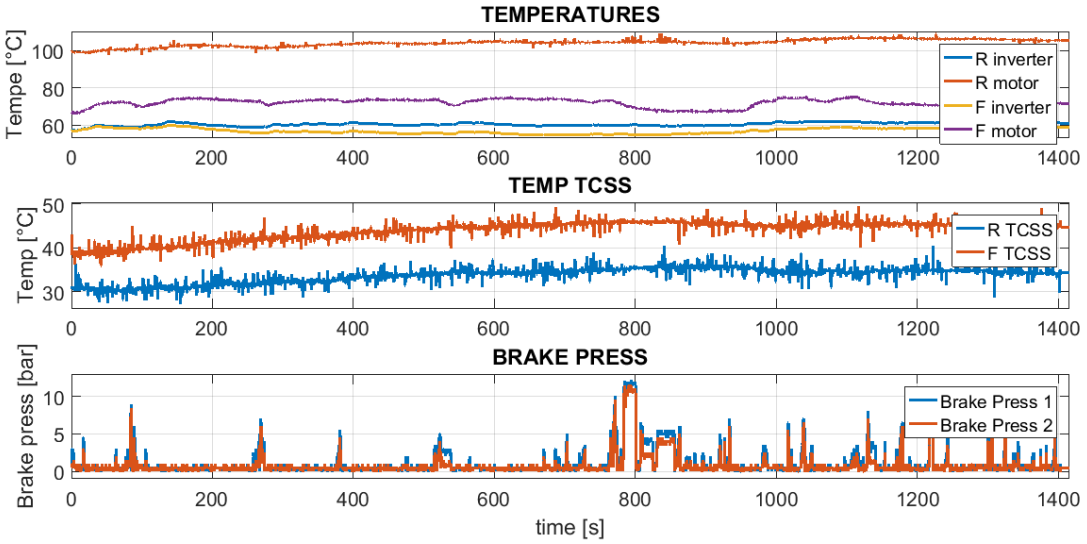


Figure 5.14 Road tests: Drive temperatures, TCSS temperatures, brake hydraulic pressures

Figure 5.14 displays measurements like the drive temperatures, the controlled differential temperatures and the hydraulic braking pressure. On the other hand, Figure 5.15 is related to internal variables of the controller which were used during tests as system state monitors.

In particular, it can be seen that the *Limit Code* for the rear drive ranges between 32 and 48. These values indicate that motor over-temperature (32) and motor + inverter overtemperature (48) limits are in effect. From Figure 5.14, in fact, the motor temperature exceeded 100 °C and the inverter temperature reached more than 60 °C, which are the respective limit intervention thresholds.

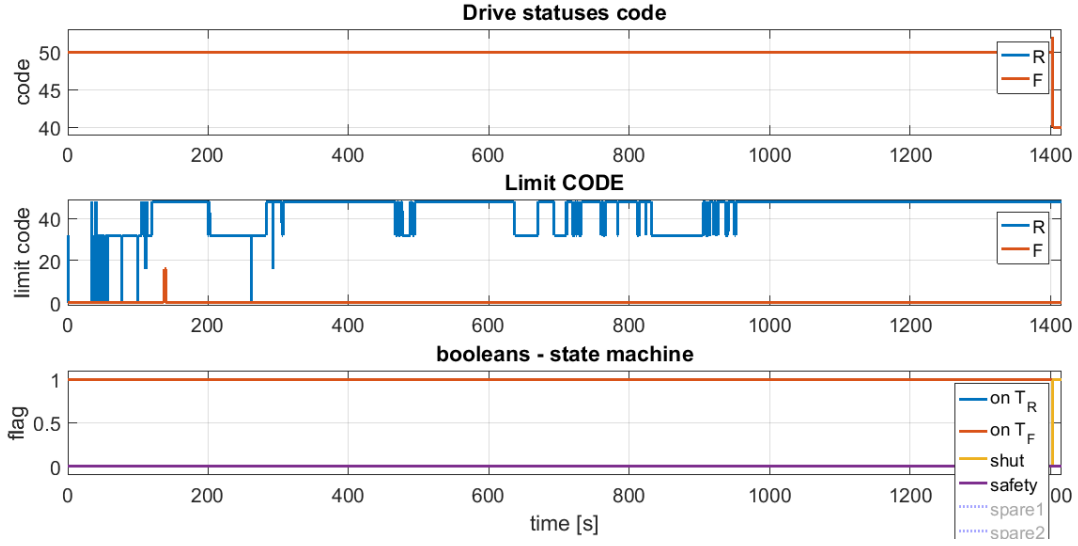


Figure 5.15 Road tests: traction management internal variables

The effect of the limit intervention can be appreciated in Figure 5.16, enlargement of Figure 5.11 and representing the trend of requested and applied torques. Considering the rear drive, the actual torque (cyan line) slightly differs from the requested one (blue line) because of the limitation.

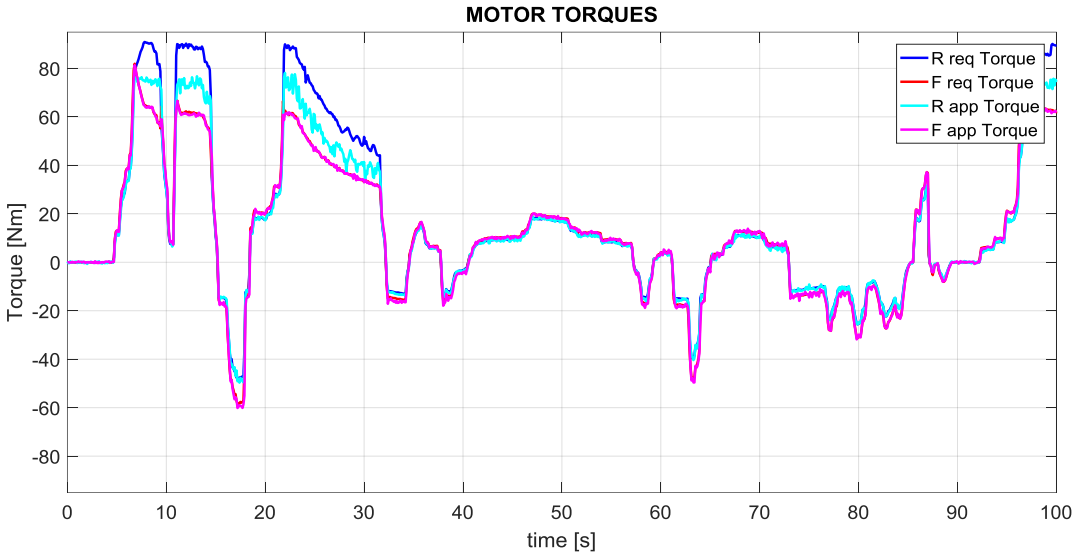


Figure 5.16 Road tests: detail of motor torques: effect of limitation

As it can be seen in Figure 5.11, the motor requested torque can be either positive or negative. This means, as explained in shakedown, that the two drives help to slow the vehicle down during braking, recovering energy that is then stored in the batteries, as seen in Figure 5.12 where the battery power becomes negative during braking.

The effect of recovery braking is better appreciated in Figure 5.17: the blue line represents the total energy required to overcome the considered driving cycle, computed by integrating the positive battery power, while the red line represents the recovered energy, as integration of the negative battery power.

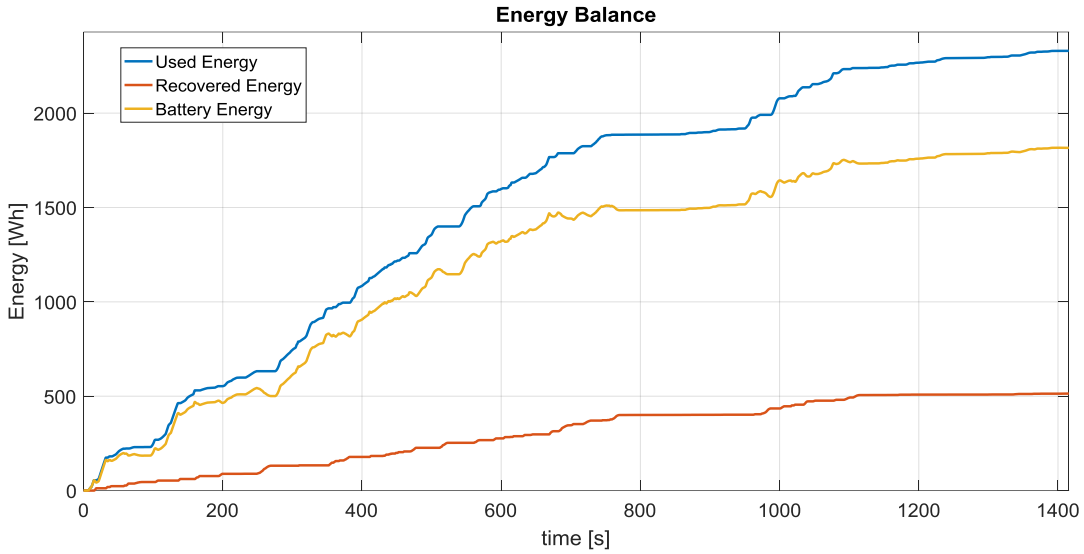


Figure 5.17 Road tests: Energy recovery during braking

It is clear that the traction management algorithm, allowing energy regeneration during braking, made it possible to save more than 20% of the total energy during real-traffic conditions.

5.7 Stability

The good dynamic behavior of the vehicle was among the AMBER-ULV project objectives and it was assessed through appropriate tests.

The typical test for evaluating the vehicle stability is the *sine with dwell* manoeuvre, defined by the ISO 19365 standard as a tool for validating the vehicle dynamic simulations. During the test, the vehicle is steered by a robot following a steering pattern of a sine wave at a frequency of 0.7 Hz with a delay of 500 ms beginning at the second peak amplitude, as shown in Figure 5.18.

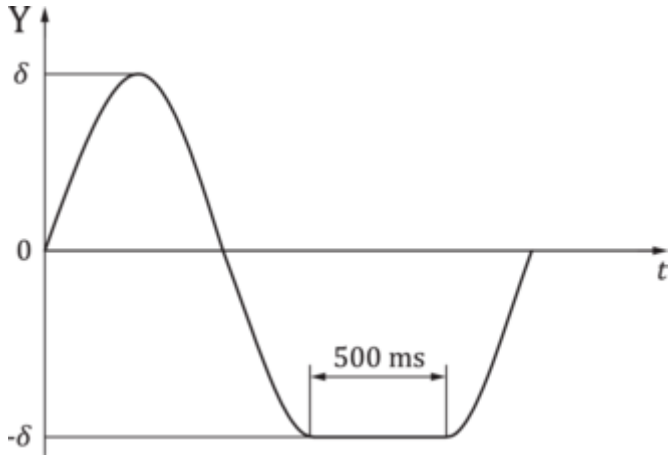


Figure 5.18 Stability test: "Sine with dwell" pattern (ISO 19365:2016)

This test was conducted on the AMBER-ULV prototype by a licensed test centre in Bollate (MI) – Italy.

The following graphs report the variables of interest for the stability evaluation through the sine with dwell manoeuvre, which mainly are the vehicle speed (Figure 5.19), the steering angle, the lateral acceleration and the yaw rate (Figure 5.20).

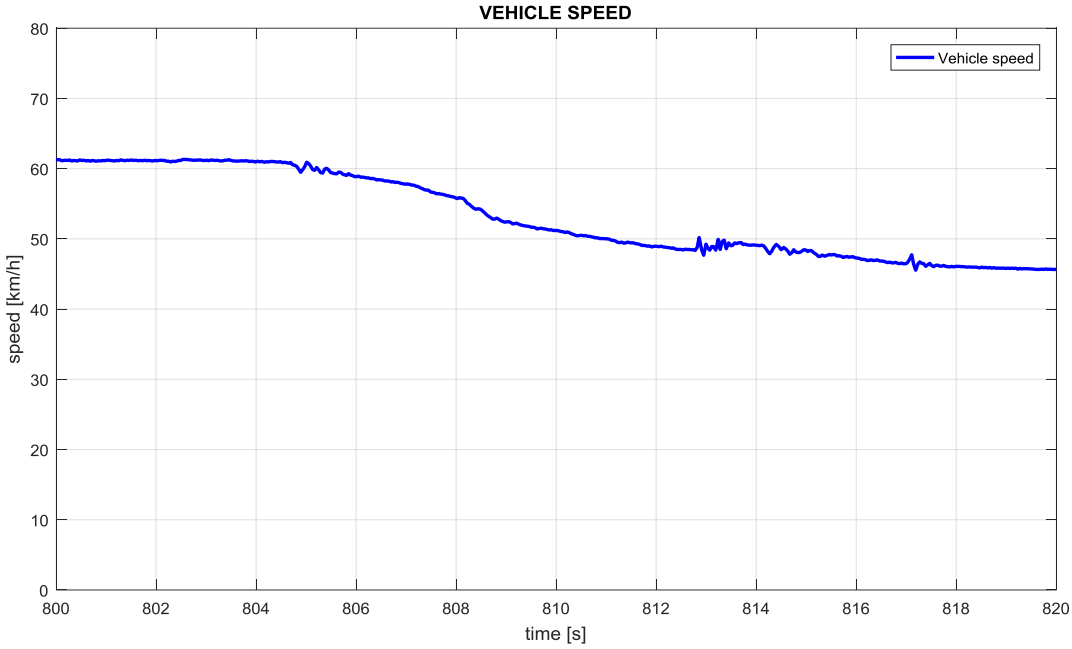


Figure 5.19 Stability test: Vehicle speed

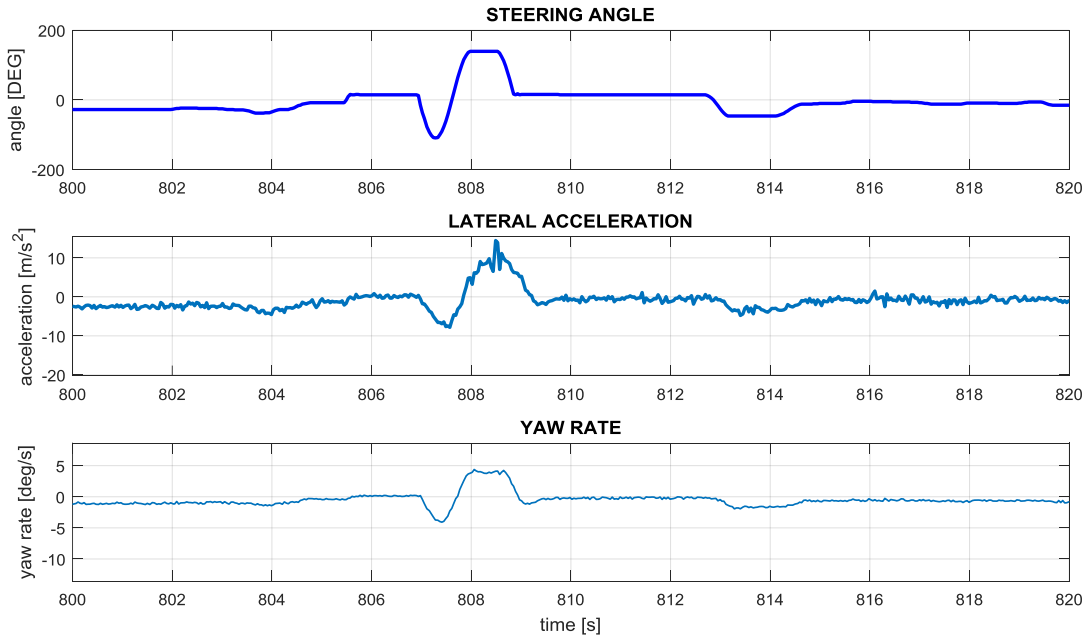


Figure 5.20 Stability test: Steering angle, lateral acceleration, yaw rate

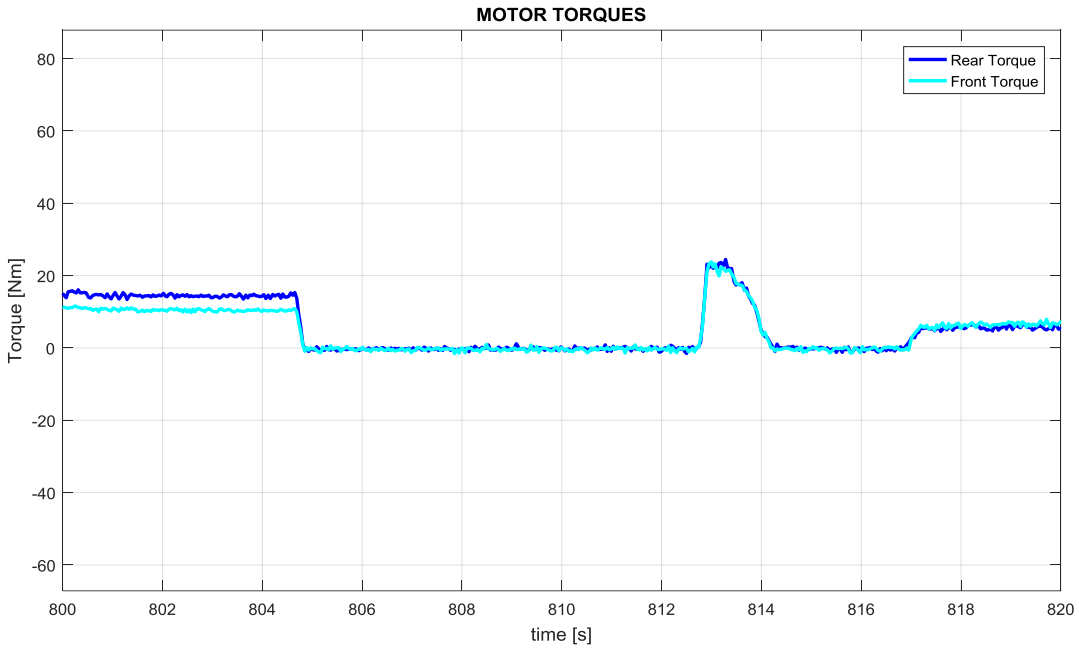


Figure 5.21 Stability test: Motor torques

This test was repeated many times, with both left and right steering for the first half of the cycle and for increasing entry speeds and steering angle amplitudes.

The passing criteria considers the yaw rate at the end of the cycle, which should be smaller than a given parameter.

The presented example refers to an entry speed of 60 km/h and a steering amplitude of 120 °, with a left steering for the first half of the cycle.

Figure 5.22 Stability test: place for "sine with dwell" manoeuvre

Conclusions

This thesis presents several aspects related to the design of a *multi-drive* traction system, in which the tractive power is divided between more than one motor.

The dissertation has general validity although, by way of example, it mainly refers to a traction system for compact electric car with two drives and two battery packs.

The first part provides analysis of the literature related to multi-drive powertrains. It describes the most common configurations and enunciate their advantages and disadvantages.

Among the benefits that are common across multi-drive powertrains, the effects of the power splitting, such as the extension of the high power operating range and vehicle dynamics enhancements, are explained. Moreover, the technical and economic feasibility of a very low voltage power system (48 V) is performed.

Because the increased complexity of the control represents the main drawback of multi-drive powertrains, a *traction management* algorithm was studied. The algorithm takes commands from the driver and generates the torque references for all the drives, while managing the state of the system with the implementation of proper protections and complying to the integration of an external stability control algorithm.

The studies conducted on multi-drive powertrains found practical application in the European project “AMBER-ULV”, for which the LEMAD laboratory developed a dual-axle, dual-drive, dual-battery traction system and its controller. In order to make the prototype work, an extensive integration activity was carried out defining the control system architecture, designing the connections with the field and writing the low-level software for interfacing signals between the controller and the vehicle, and manage high-level algorithms.

The *model-based design* approach was exploited to effectively develop the control system and to automatically generate the code for the microprocessor starting from the aforementioned *traction management* algorithm.

The completed prototype was extensively tested in several conditions, aiming to demonstrate the validity of the studies presented in this dissertation.

For the good outcome of the tests, a telemetry system was devised and deployed. It allowed to effectively monitor the vehicle parameters during tests and record them for post-processing purposes.

Examples of the conducted tests, which include accelerations, energy consumption estimation and real-traffic runs, are presented in this thesis.

The proposed approach for the development of multi-drive systems allows realizing electric powertrain for cars with the following characteristics:

- Higher reliability of the electric driveline, due to the intrinsic redundancy characteristics of the system and to the capability of the control system to operate the powertrain even in case of component faults.

- Possibility to realize a full electric powertrain for compact cars using inverter with limited *kVA* ratings, sized for a fraction of the whole traction power. This characteristic enables the possibility to realize a full electric car using a voltage level for the power system of 48 V.
- Possibility to enhance the road handling of the vehicle by combining the traction system with the mechanical brake control implemented in the active stability system (ESC).

The listed outcomes of the project are paired with the reduction of the system cost, mainly due to the use of smaller size and simpler technology components. In this way the proposed approach for realizing the control system of multi drive, full electric powertrains facilitates the implementation of low-voltage, multi-drive system.

The results of this Ph.D. work could significantly contribute to reducing the cost of electric powertrain, yielding to increase the spreading of electric vehicles.

References

[1] E. Chemali, M. Peindl, P. Malysz, and A. Emadi, “Electrochemical and Electrostatic Energy Storage and Management Systems for Electric Drive Vehicles: State-of-the-Art Review and Future Trends,” *IEEE J. Emerg. Sel. Top. Power Electron.*, 2016.

[2] J. M. Timmermans *et al.*, “Batteries 2020 - Lithium-ion battery first and second life ageing, validated battery models, lifetime modelling and ageing assessment of thermal parameters,” *2016 18th Eur. Conf. Power Electron. Appl. EPE 2016 ECCE Eur.*, 2016.

[3] “AMBER-ULV (604766) Grant agreement - Annex 1: ‘Description Of Work,’” 2013.

[4] S. Cui, S. Han, and C. C. Chan, “Overview of multi-machine drive systems for electric and hybrid electric vehicles,” *Transp. Electrif. Asia-Pacific (ITEC Asia-Pacific), 2014 IEEE Conf. Expo*, no. Mcm, pp. 1–6, 2014.

[5] V. Agarwal and M. Dev, “Introduction to hybrid electric vehicles: State of art,” *Eng. Syst. (SCES), 2013 Students Conf.*, pp. 1–6, 2013.

[6] C. Geng, L. Mostefai, M. Denai, and Y. Hori, “Direct yaw-moment control of an in-wheel-motored electric vehicle based on body slip angle fuzzy observer,” *IEEE Trans. Ind. Electron.*, pp. 1411–1419, 2009.

[7] L. De Novellis *et al.*, “Direct yaw moment control actuated through electric drivetrains and friction brakes: Theoretical design and experimental assessment,” *Mechatronics*, vol. 26, pp. 1–15, 2015.

[8] Q. Lu *et al.*, “Enhancing vehicle cornering limit through sideslip and yaw rate control,” *Mech. Syst. Signal Process.*, vol. 75, pp. 455–472, 2016.

[9] L. De Novellis, A. Sorniotti, and P. Gruber, “Driving modes for designing the cornering response of fully electric vehicles with multiple motors,” *Mech. Syst. Signal Process.*, 2015.

[10] Y. Li, G. Yin, X. Jin, C. Bian, and J. Li, “Impact of Delays for Electric Vehicles With Direct Yaw Moment Control,” *J. Dyn. Syst. Meas. Control*, 2015.

[11] C. Rossi, D. Pontara, M. Bertoldi, and D. Casadei, “Two-motor , two-axle traction system for full electric vehicle,” *EVS29 Int. Batter. Hybrid Fuel Cell Electr. Veh. Symp.*, 2016.

[12] A. Patzak, F. Bachheibl, A. Baumgardt, G. Dajaku, O. Moros, and D. Gerling, “Driving range evaluation of a multi-phase drive for low voltage high power electric vehicles,” *2015 Int. Conf. Sustain. Mobil. Appl. Renewables Technol. SMART 2015*, 2016.

[13] F. Bachheibl and D. Gerling, “High-current, low-voltage power net,” *2014 IEEE Int. Electr. Veh. Conf. IEVC 2014*, 2015.

[14] M. Orehek and C. Robl, “Model-based design of an ECU with data- and event-driven parts using auto code generation,” *Proc. 2001 ICRA. IEEE Int. Conf. Robot. Autom. (Cat. No.01CH37164)*, pp. 1346–1351, 2001.

[15] M. Ahmadian, Z. J. Nazari, N. Nakhaee, and Z. Kostic, “Model based design and sdr,” *2nd IEE/EURASIP Conf. DSP Enabled Radio*, 2005.

[16] W. Li, G. Xu, H. Tong, and Y. Xu, “Design of Vehicle Control Unit Based on DSP for a Parallel HEV,” *2007 IEEE Int. Conf. Autom. Logist.*, pp. 1597–1601, 2007.

[17] M. Hu, Y. Huang, C. Zhao, X. Di, B. Liu, and H. Li, “Model-based development and

automatic code generation of powertrain control system," *2014 IEEE Conf. Expo Transp. Electrif. Asia-Pacific (ITEC Asia-Pacific)*, 2014.

[18] A. Wagener, C. Koerner, and H. Kabza, "Simulation-Based Automatic Code Generation for ECUs in Distributed Control Systems, Applied in a Testbed for a Hybrid Vehicle Drivetrain," *IEEE Int. Symp. Ind. Electron.*, pp. 643–648, 2000.

[19] B. Vogel-Heuser, D. Schütz, T. Frank, and C. Legat, "Model-driven engineering of Manufacturing Automation Software Projects - A SysML-based approach," *Mechatronics*, vol. 24, no. 7, pp. 883–897, 2014.

[20] J. Kang, S. Jin, and W. Lee, "Developing Software of Electronic Throttle Controller using Automatic Code Generation Technique," *SICE-ICASE Int. Jt. Conf.*, pp. 4393–4397, 2006.

[21] P. M. Menghal and a J. Laxmi, "Real time control of electrical machine drives: A review," *Power, Control Embed. Syst. (ICPCES), 2010 Int. Conf.*, 2010.

[22] S. E. Viswanathan and P. Samuel, "Automatic code generation using unified modeling language activity and sequence models," *IET Softw.*, Jul. 2016.

[23] M. Yamazaki, S. Sureshbabu, M. Loftus, R. Crandell, and M. Brackx, "Analysis of automatically generated vehicle system control software in a HIL environment," *Proc. Am. Control Conf.*, vol. 4, pp. 3135–3140, 2002.

[24] A. Ingalalli, H. Satheesh, and M. Kande, "Platform For Hardware In Loop Simulation," *2016 Int. Symp. Power Electron. Electr. Drives, Autom. Motion*, 2016.

[25] "High CPU usage when reading serial port." [Online]. Available: http://forums.ni.com/.

[26] "Best Practices for DMA Applications (FPGA Module)." [Online]. Available: http://zone.ni.com/.

Publications

C. Rossi, D. Pontara and D. Casadei, “e-CVT Power Split Transmission for Off-Road Hybrid-Electric Vehicles,”. Proc. of 2014 IEEE Vehicle Power and Propulsion Conference (VPPC), IEEE The Institute of Electrical and Electronics Engineers, Inc., Coimbra (Portugal), 27 - 30 October 2014.

S. Schiavitti, C. Rossi and D. Pontara, “An innovative Energy Management System acting on multiple battery packages,”. Proc. of 16th Stuttgart International Symposium on Automotive and Engine Technology, Stuttgart (Germany), March 15-16, 2016.

C. Rossi, M. Alirand, A. Galli, P. Phiani and D. Pontara, “Achievements with Model-based Development on the Innovative traction and stability system of the AMBER-ULV Car,”. Proc. of 16th Stuttgart International Symposium on Automotive and Engine Technology, Stuttgart (Germany), March 15-16, 2016.

C. Rossi, D. Pontara, M. Bertoldi, D. Casadei, “Two-motor, Two-axle Traction System for Full Electric Vehicle,” Proc. of EVS 29, Electric Vehicle Symposium & Exhibition, Montreal (Canada) June 19-22 2016.

AMBER-ULV Project Deliverables

M. Alirand, D Pontara, P. Magnin: “Preliminary vehicle virtual model,”. AMBER-ULV Project Deliverable D4.1, November 2014.

C. Rossi, Y Gritli, D. Casadei, F. Filippetti, D. Pontara: “Motor Control Algorithm,”. AMBER-ULV Project Deliverable D3.3, January 2015.

C. Rossi, D. Pontara, M. Bertoldi: “Unified Hardware Control Board,”. AMBER-ULV Project Deliverable D4.4, September 2016.

A. Galli, D. Pontara, M. Bertoldi, S. Gerstmayr: “Electric/Hydraulic Brake Blending,”. AMBER-ULV Project Deliverable D6.2, August 2016.

A. Galli, S. Gerstmayr, O. Mannuß, D. Pontara, M. Bertoldi: “AMBER-ULV Demonstrator Results,”. AMBER-ULV Project Deliverable D7.2, August 2016.