Battery Management System Design

Advanced Automotive Electronic Engineering

Authors:
Anna Daniela Delli Noci
Carlo Edoardo Malaspina
Michele Gazzarri
Mirko Shytlla

Professors:
Carlo Concari
Alessandro Soldati

June 2, 2021

MOTORVEHICLE UNIVERISITY OF EMILIA ROMAGNA

Master degree: Advanced Automotive Electronics Engineering

Battery Management System Design

Abstract

In the following paper are reported the steps that led to the realization of a Battery Management System. The central core of the activity is applying a Model-Based design in order to simulate and implement the controller in a chosen environment. This approach is well underlined among the chapters that articulate this paper: first it is designed the plant of a battery pack which is able to simulate as faithful as possible the behaviour of a real system. After this it has been implemented a controller of the pack itself. The design of the model, its simulation and the hardware setup for a correct implementation are performed in Simulink environment.
At the end, after the overall simulation, the controller was implemented on real hardware and its behaviour was compared to the one that was previously simulated.

Contents

Abstract iii

1 Introduction 1

2 BMS: Plant 3
2.1 Plant Model . . . 3
2.2 Battery Pack . . . 3
Battery Module . . . 4
Battery Segment . . . 5
2.2.1 Single Cell . . . 6
Passive Balancing . . . 7
2.3 Contactor Module . . . 8
2.4 Battery Charger and Drivetrain . . . 10

3 BMS: Controller 13
3.1 Contactor Logic . . . 14
3.2 SOC Estimation . . . 15
3.2.1 Coulomb counting . . . 16
3.2.2 Extended Kalman Filter . . . 16
3.3 Balancing Logic . . . 17
3.4 Fault Monitor . . . 18

4 Testing 21
4.1 Model-in-the-loop (MIL) . . . 23
4.1.1 Simulation of MIL . . . 24
Simulation pattern . . . 24
4.1.2 Precharge and Discharge phase . . . 25
Fault Injection and Cell Balancing . . . 26
State of Charge estimation . . . 28
Over-Current warning . . . 29
4.2 Software-in-the-loop (SIL) and Processor-in-the-loop (PIL) . . . 29

5 Conclusions and further improvements 33

A Battery Parameters 35

B Contactor Parameters 37

C Charger and Drivetrain Parameters 39

D Fault Parameters 41

List of Abbreviations

BEV	**B**attery **E**lectric **V**ehicle
BMS	**B**attery **M**anagement **S**ystem
EKF	**E**xtended **K**alman **F**ilter
FSM	**F**inite **S**tate **M**achine
MIL	**M**odel **I**n the **L**oop
OCV	**O**pen **C**ircuit **V**oltage
PHEV	**P**lug-in **H**ybrid **E**lectric **V**ehicle
PIL	**P**rocessor **I**n the **L**oop
SIL	**S**oftware **I**n the **L**oop
SOC	**S**tate **O**f **C**harge
SOH	**S**tate **O**f **H**ealth
UI	**U**ser **I**nterface

Chapter 1

Introduction

Modern battery systems, especially the ones based on lithium-ion cells, require careful monitoring and control of state of each individual cell to ensure safety of operations, good performance and durability.
This crucial task is assigned to the *Battery Management System* (BMS) which monitors and controls each cell, such that small tolerances are allowed with respect to strict operating and environmental conditions. Indeed, voltages provided by single lithium-ion cells are much lower than the one needed in order to drive Plug-in Hybrid Electric Vehicle (PHEV) or Battery Electic Vehicle (BEV).
To cope with this problem, the current solution consists in connecting in series hundreds of cells to construct battery modules and packs that could fulfill the needs of the previously mentioned PHEV or BEV. However as a consequence of the connection in series of a huge number of cells, a trivial phenomena called *cell-to-cell state of charge (SOC) unbalancing* arises.
The physical reasons can be identified in:

- Cell manufacturing errors;
- Thermal management.

The SOC unbalancing could lead to over-charging or over-discharging conditions as well as fast-aging, implying that a control is needed in order to avoid all the previously listed conditions.
All these phenomena lead to a schematic approach that provides a first division in two main blocks, indeed they are:

- Controller;
- Plant.

This division is crucial since it highlights how the plant is responsible of simulating all the capabilities and physical effects of a real battery pack. On the other hand, there is the software that is able to manage and handle in a proper way all the manipulation needed for a correct operation.
Each of them will be deeply investigated in the next sections 2 and 3, however it could be appreciated to firstly have a general and brief overview on their main features.

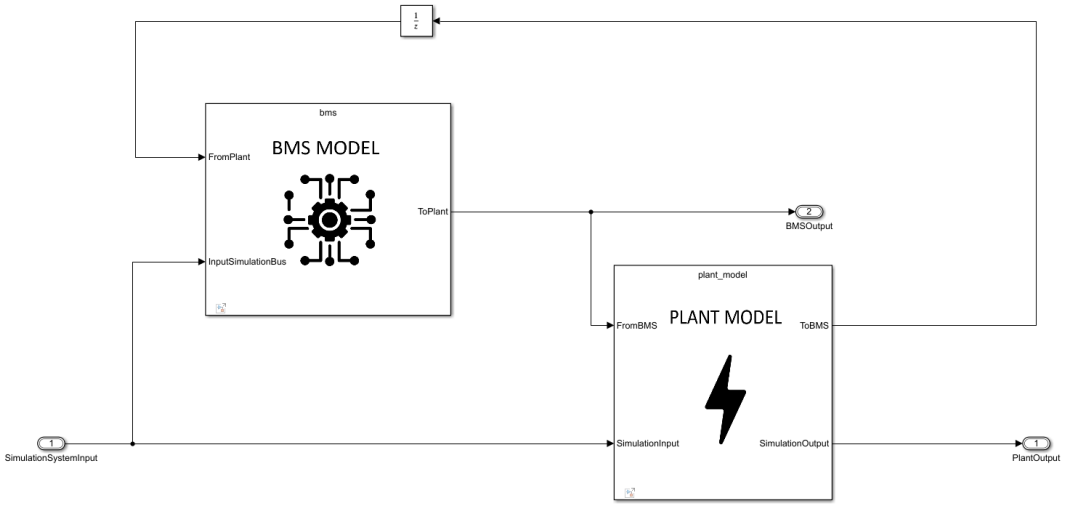

FIGURE 1.1: System model

The *Controller* is responsible to handle all the logic parts and algorithms, such that an overall system correct and safe behavior is guaranteed. Indeed all the control signals provided are then further elaborated and managed by the *Plant* block, that consists in the model of the battery pack and its associated circuitry and peripherals.

After a general description of all the physical model and the control strategies implemented, several simulations are performed in order to verify the overall working and to evaluate the performances of the chosen design.
Thanks to those simulations it is also possible to identify and stress some of the possible upgrades that may improve the functionalities and performances of this configuration.

The very last step performed has been the implementation of the controlling software on real hardware. Using Simulink supporting tools, it has been possible to move from a simulation environment to a real-world one.
First of all, C code was generated targeting an STM32 micro-controller. Then, exploiting PIL add-on, it was possible to compare the behaviour of the system running on the micro-controller with respect to the simulated one.

Chapter 2

BMS: Plant

2.1 Plant Model

In the present section, there will be a careful focus on how the Plant block has been realized and which are all the elements that have been modeled in order to get its final design. As it can be appreciated in Fig. 2.1 within the Plant Model, three different main sections can be identified:

- Battery Pack;
- Contactor Module;
- Battery Charger and Drivetrain.

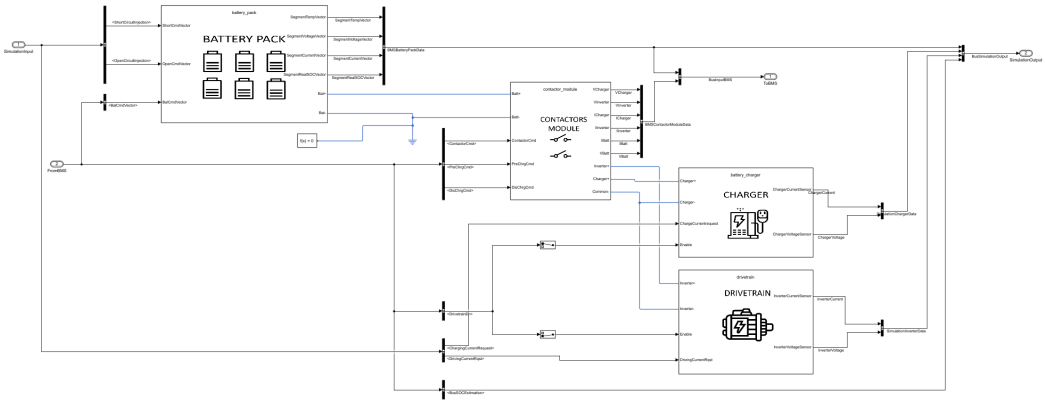

FIGURE 2.1: Plant model.

2.2 Battery Pack

The battery pack, that is the first element starting from the left comparing in the Plant Model, has been modeled such that three different battery modules are connected in series as depicted in Fig. 2.2 .

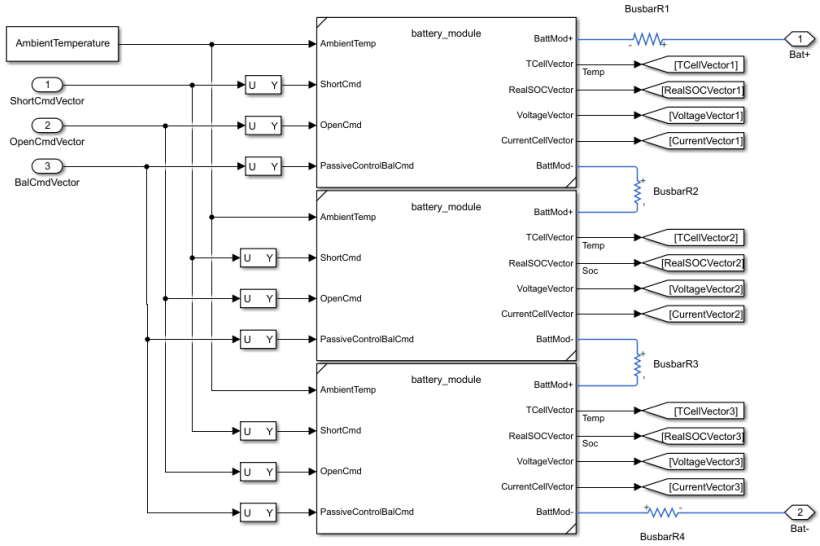

FIGURE 2.2: Battery pack model.

Each battery module has in input four different signals coming both from the controller algorithms and from the simulation environment. They will be listed in the following:

- **AmbientTemp:** parameters that describe the initial ambient temperature of the simulation. Can be modified through the *system_setup.m* file;
- **ShortCmd:** [1x10] boolean array that can be used to inject a single cell short circuit fault from the simulation environment to test the controller behaviour;
- **OpenCmd:** [1x10] boolean array that can be used to inject a single cell open circuit fault from the simulation environment to test the controller behaviour;
- **PassiveControlBalCmd:** [1x10] boolean array used to activate the passive balancing circuit of each cell;

It can be noticed from Fig. 2.2, that each battery module is electrically connected through two resistors (both from + and - terminals) whose aim is to model the busbars resistance of a real situation. Also in this case, the value of each resistance can be modified from *system_setup.m*.

Battery Module

By further expanding the previous explained Battery Pack, it is possible to have a look on how the Battery Module has been realized. As shown in Fig. 2.3 there are two main parts connected together which are:

- Battery Segment;
- Passive Balancing.

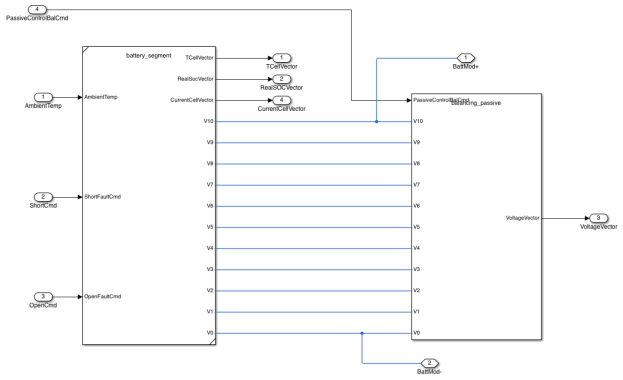

FIGURE 2.3: Battery module model.

Battery Segment

The first element present in the Battery Module starting from the left is the Battery Segment. It is depicted in Fig. 2.4, and it is composed of 10 cells connected in series and described in the next section.

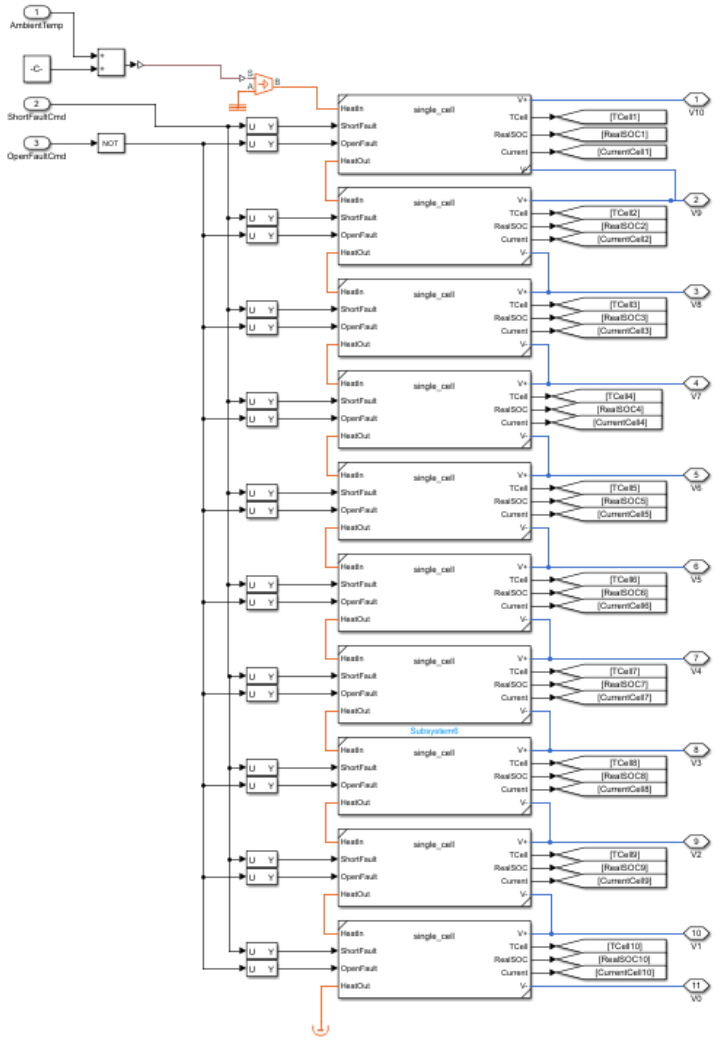

FIGURE 2.4: Battery segment model.

2.2.1 Single Cell

The model used to describe the behaviour of every cell of the battery pack, depicted in Figure 2.5, is based on **Battery (Table-Based) with Thermal Port** block from the *Simscape Electrical* library. Thanks to this block, it has been possible to choose an already parameterized cell model that is named **Panasonic: NCR18650BD**.

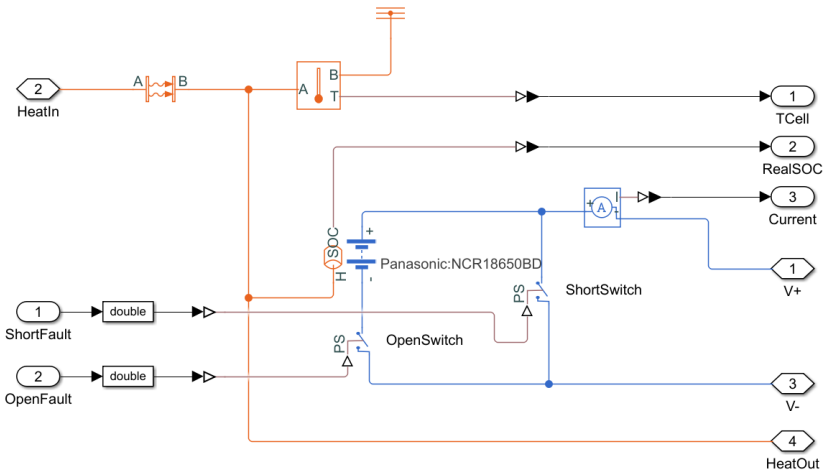

FIGURE 2.5: Single cell model.

The cell behavior has been modeled both from electrical and thermal point of view. Specifically, on the top left part of Figure 2.5, a **Convective Heat Transfer** block has been used to model the thermal interaction between cells, supposing that they are placed inside the segment with a small air gap between them, and considering only the first cell to exchange heat with ambient. Also in this case, the cell area and the convective heat transfer coefficients can be modified from the setup file.

Passive Balancing

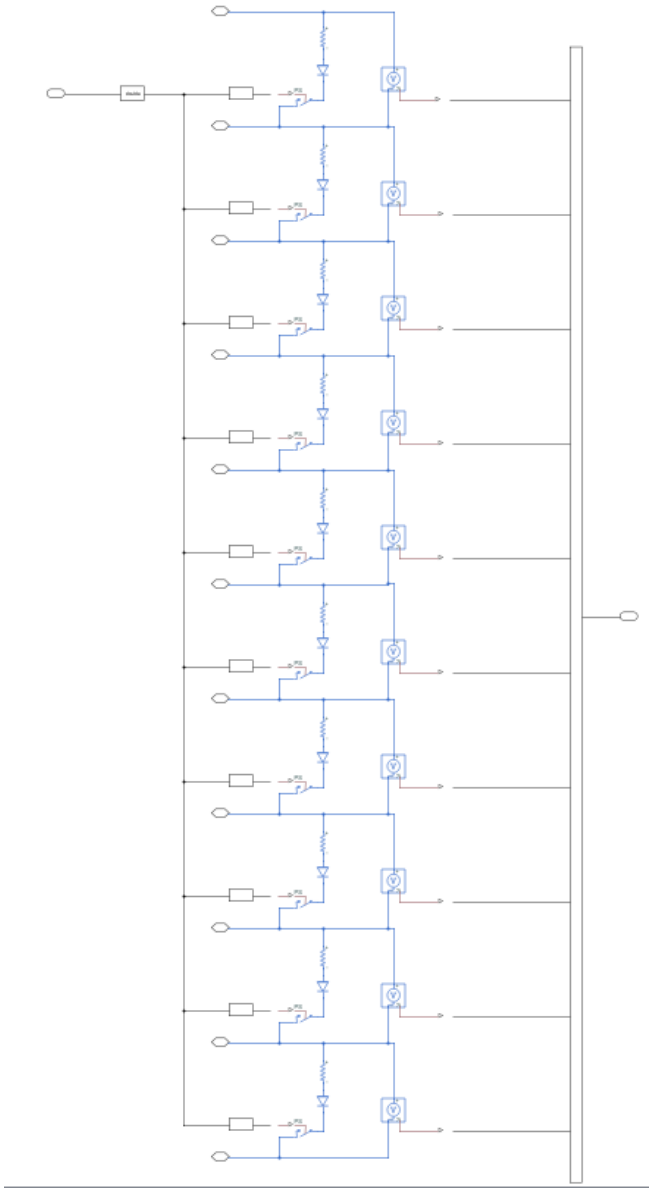

FIGURE 2.6: Passive Balancing Circuit

It was already mentioned the fact that a passive logic, as well as passive circuitry is needed in order to ensure balancing among each cell that compose the battery segment. In Fig. 2.6 it is possible to appreciate the model implemented.

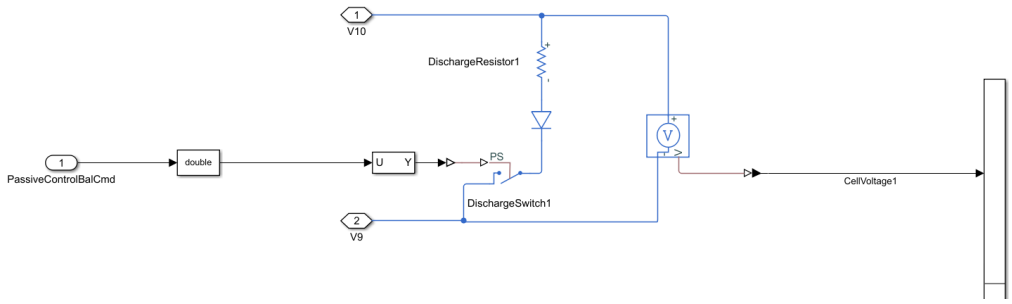

FIGURE 2.7: Passive Balancing Circuit detail.

Each switch is selectively controlled and closed each time that the corresponding cell to which it is connected, has the need to be discharged. The discharge of every cell is performed through a parallel resistor whose value can be selected from the setup file.
This process allows to lower the *SOC* of the cell identified, in order to maintain the same SOC among cells belonging to same segment. One of the main benefits of this strategy is the maximization of the energy stored inside the battery pack.

2.3 Contactor Module

When dealing with high power applications, three main devices have to be placed in between the battery pack and the powertrain:

- Contactors;
- Pre-charge circuit;
- Post-discharge circuit.

A contactor is essentially a relay for very high power applications, characterized by the fact that it can easily handle hundreds of ampere without blowing. Unlike relays, contactors are designed with features to control and suppress the arc produced when interrupting highly inductive loads. They are the most suitable devices for disconnecting and isolating the battery pack from the rest of the drivetrain circuits.
In addiction to this, it is important to consider the possibility of having the need to drive high capacitive loads. Usually, some large capacitors are placed both at the input of the inverter, as well as, at the output stage of the battery chargers. Their aim is to filter high voltage spikes during transient of high current draw, thus preventing an high harmonics content in the main DC link. For this reason, the need of a pre-charge resistor arises. In order to preserve the contactor life, it is important to limit the in-rush currents due to the capacitor charging phase. The current flowing through a capacitor during its charge process its known to be:

$$I = C\frac{dV}{dt} \tag{2.1}$$

During transient, the derivative component of the Equation 2.1 tends to $+\infty$, hence pushing the current to $+\infty$ too. As a reference, in Table 2.1, it is reported an example of in-rush currents at different time steps and voltage levels, for a 11000 μF capacitor.

Battery Voltage	Peak in-rush current			
	1 ms	10 ms	100 ms	1 s
28V	310 A	31 A	3.1 A	0.31 A
610V	6710 A	671 A	67 A	7 A

TABLE 2.1: Peak in-rush current

Given these very high current values, it is clear that some limiting circuit is needed. A common pre-charge circuit is reported in Figure 2.8. It is composed by an high power resistor and a relay. During its operation, the current flows through the resistor and charges the capacitor with a certain time constant, depending on both the capacitor and the resistor values.

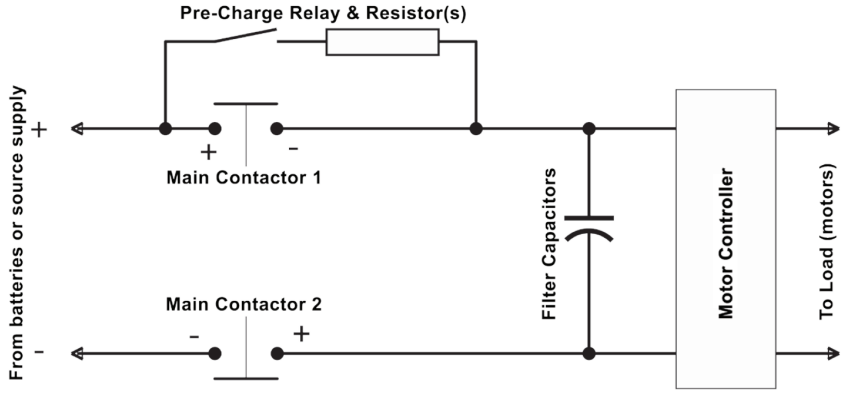

FIGURE 2.8: Pre-Charge circuit

By measuring the capacitor value, it is possible to tune the resistor value in order to obtain a certain pre-charge time constant.

$$\tau = RC \tag{2.2}$$

Finally, for safety reasons, two post-discharge resistors have been added in order to rapidly discharge the capacitors and to ensure that each time that the contactors are disconnected there is no voltage at the DC links. In Figure 2.9 is reported the complete implementation of the contactor module, in which it is possible to find the three contactors as well as the pre-charge and post-discharge resistors.

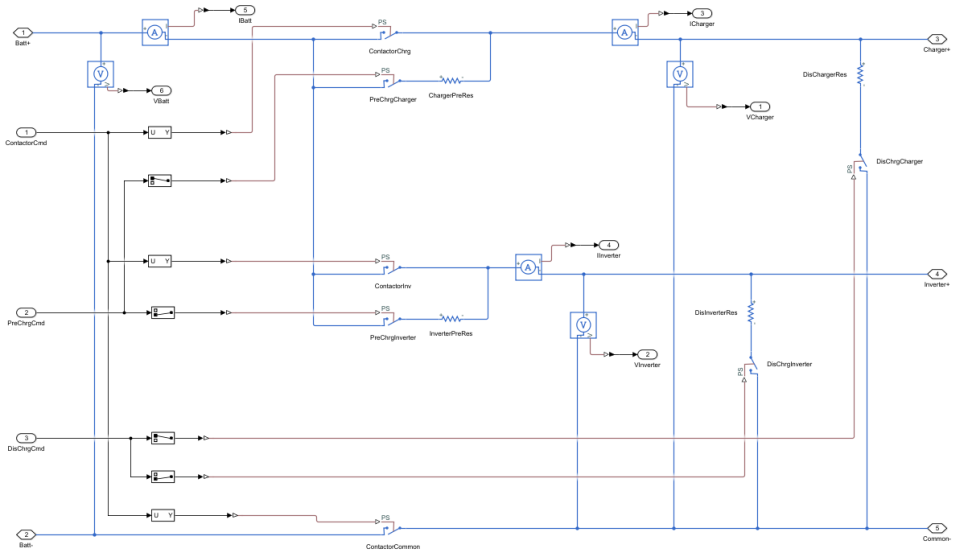

FIGURE 2.9: Contactor module

2.4 Battery Charger and Drivetrain

In this section two components external to the battery pack, but of key importance for the simulation are discussed:

- **Drivetrain**;
- **Battery Charger**.

These two models are quite similar, exception made for the sign of the current source used inside them. For this reason, only the Drivetrain model is reported in Figure 2.10.

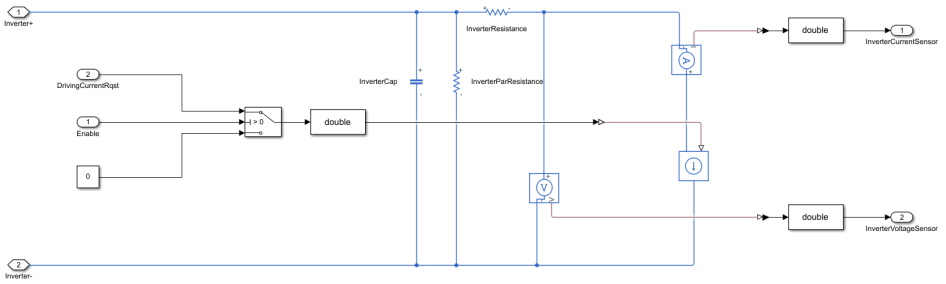

FIGURE 2.10: Drivetrain model

The drivetrain system has the main goal of modeling the other parts of an electric powertrain exception made for the battery pack. Aforementioned parts are the electric motor and the motor controller/inverter. Typically the battery pack is directly connected to the inverter that sinks a certain amount of current from it, depending from several parameters like torque request, motor speed, inverter and motor temperature, battery pack SOC and so on.
The input of this subsystem are:

- **Inverter+ and Inverter-**: PMC Port terminals that must be connected to the Contactor Module described in previous Section;

- **Enable**: This signal is generated from the BMS Controller and allow the Drivetrain system to sink a current larger than 0 from the battery pack;
- **DrivingCurrentRqst**: Amount of current requested from the battery pack in Ampere. It is typically possible to convert this current request to a Torque Request applying a gain accordingly to the machine parameters of the motor and of the inverter.

The inverter and the battery charger have been modeled considering a series resistance, a parallel resistance and an input capacitance, in addition to a Controlled Current Source that is responsible for the current requested and absorbed. In addition to these elements, also an Amperometer and a Voltage Sensor have been added and their measurements are reported as output of the system.

Chapter 3

BMS: Controller

The most important part of a Battery Management System is the controller itself. It is involved in monitoring the operating conditions, estimates the battery states and actuates different strategies based on the aforementioned parameters. In the very first stage of the project development, a list of required features has been developed in order to better organize the workload and split it among the group members. As a result, here are reported the main characteristics that the controller must have:

- Monitors voltage, temperature and current of each cell;
- Manages the activating and deactivating sequence for the contactors as well as for the pre-charge and post-discharge resistors;
- Creates alarms in case of different kind of fault that may arise;
- Estimates battery pack state-of-charge (SOC) and state-of-health (SOH);
- Prevents safety critical conditions such as over discharge, over charge or over temperature;
- Provides a cell balancing method to equalize different cell voltages to the same level.

Starting from these requirements, all of the different sub-modules that have been designed are presented in details in the next sections.

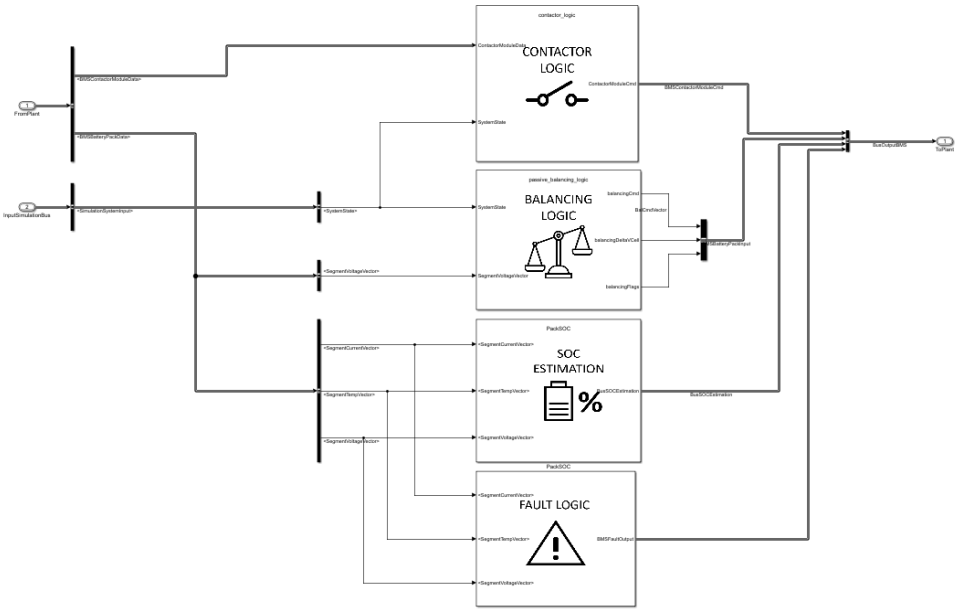

FIGURE 3.1: Controller Block

3.1 Contactor Logic

As described in Section 2.3, the battery pack is connected to the external environment through three different contactors: the first for the positive terminal of the battery charger, the second for the positive terminal of the inverter and the last one for the common negative terminal. That way, in case of emergency, the battery pack can be completely disconnected from the rest of the vehicle. Moreover there are some additional relays needed for the pre-charge or the post discharge processes. These are needed in order to limit the in-rush currents that will flow in case of connecting highly capacitive loads.
The contactor logic block is responsible of monitoring the global *SystemState* variable. In case of transitions and based also on the three main voltage levels, it will select the corresponding strategy, thus ensuring a proper turn-off or turn-on sequence of all the different switches.

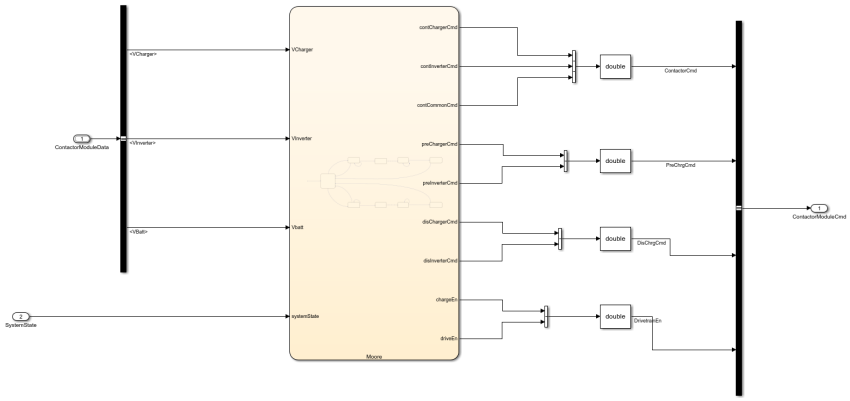

FIGURE 3.2: Contactor Logic

In order to have a better understanding, there will be a focus on its Finite State Machine. As it can be appreciated in Figure 3.3, starting from the left there is the entry point with an initialization state in which all the variables are set to their default value, then the FSM is split in two similar branches with four different states each. Each branch is in charge of controlling one of the two drivetrain components: the one above is for the charger while the one below is for the inverter.

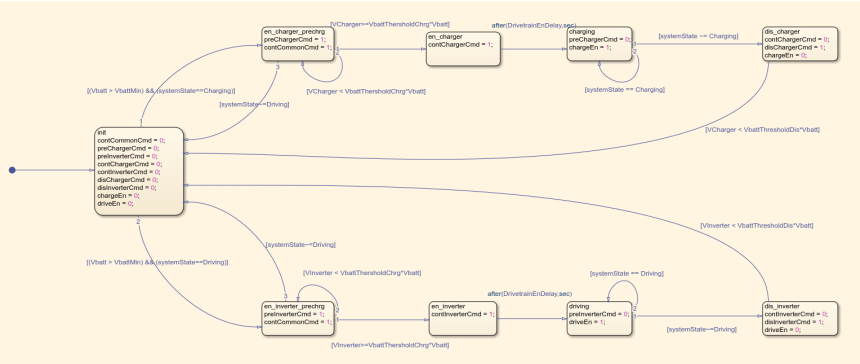

FIGURE 3.3: Contactor Logic Finite State Machine

When the *SystemState* changes from "Parking" to either "Driving" or "Charging", the corresponding branch is run from left to right. At the beginning, the pre-charge resistor's relay is turned on, allowing the current to flow in a supervised way until

the target voltage is reached. At this point, the FSM switches to the second state in which the contactor is turned on. After a user-programmable delay the pre-charge resistor is turned off and a drivetrain enable flag is set. The FSM maintains its state until the *SystemState* is changed again. As soon as this happens the last block is involved, in which the contactor is switched immediately off and the post-discharge resistor's relay is turned on. As before, when the voltage of the output rail reaches the desired level, the system goes back to the initialization position, waiting to restart from the beginning. Since the balancing condition does not involve the actuation of any switch in the contactor module, it is not considered in this framework. In Figure 3.4 are reported the output voltage and current waveforms generated by a transition of the *SystemState* from "Parking" to "Driving".

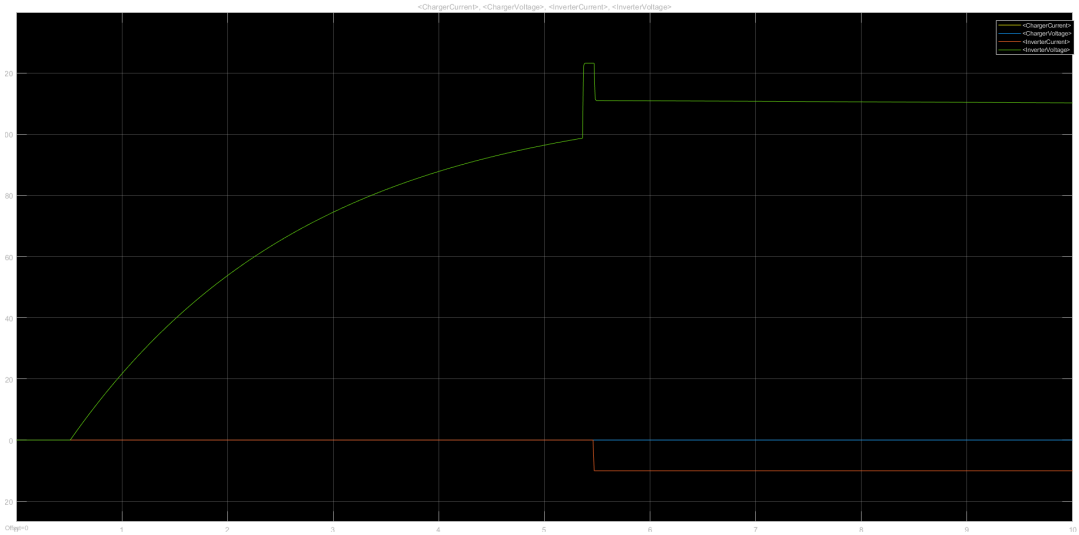

FIGURE 3.4: Inverter Voltage and Current waveforms from "Parking" to "Driving"

3.2 SOC Estimation

Among all the information meaningful that can be obtained from a battery pack, State of Charge estimation is one of the most critical.
However, unlike other quantities such as voltage or temperature, SOC cannot be measured directly from a particular sensor. Because of this limit it has to be estimated processing some of the information that are already available and measured by some of the on-board sensors.
Although battery charge and discharge are affected by several factors, most of the estimation techniques are based on the investigation of three particular parameters:

- Voltage of the pack;
- Current absorbed or released by the pack;
- Temperature of the pack.

Designing a proper sensor fusion of these three quantities allows a good-quality estimation of the SOC.
In this particular implementation two different methods have been studied:

- Coulomb counter;
- Extended Kalman Filter.

3.2.1 Coulomb counting

Coulomb counting method deploys an algorithm that is quite simple and does not require high computational power. It measures the discharging current of the battery and integrates it over time in order to estimate the SOC.
The simplicity and very low computational cost although carry some limitations:

- Diverging estimation if errors in current measurement occur;
- Impossibility to recover from a wrong initial condition.

The accuracy can be improved by weighting the discharging or charging current over some effects that may impact the estimation like, for example, the temperature of the battery pack. In Figure 3.5 it is shown the estimator deployed in our plant.

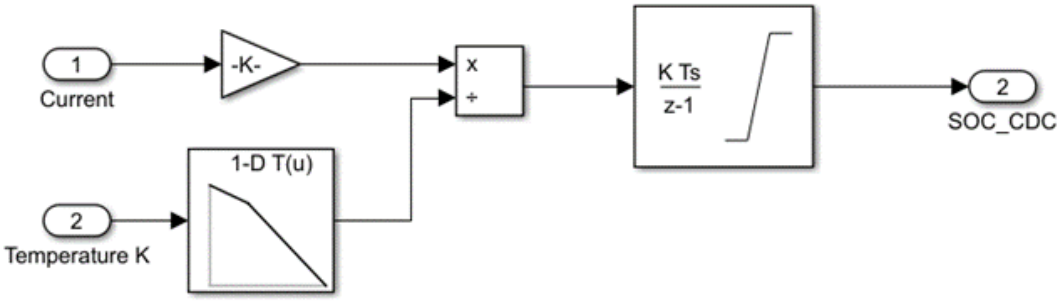

FIGURE 3.5: Coulomb Counter Estimator

3.2.2 Extended Kalman Filter

This sensor fusion technique overcomes the limitations of the so-called book-keeping methods. An adaptive system like Kalman filter for example provides a powerful tool that performs accurate estimations of SOC thanks to real-time state estimation. This adaption to changing conditions is achieved thanks to a Model-Based design of the plant that has to be observed and thanks to a fusion of different information such as voltage of the pack, its temperature and current absorption or release.
Despite the precision advantage, its tuning and setup are more complicated than the simple Coulomb Counting. Moreover, in a real-time application the resources needed can be very high. Because of this, the observer has been designed but its tuning and implementation is pointed as a future upgrade.

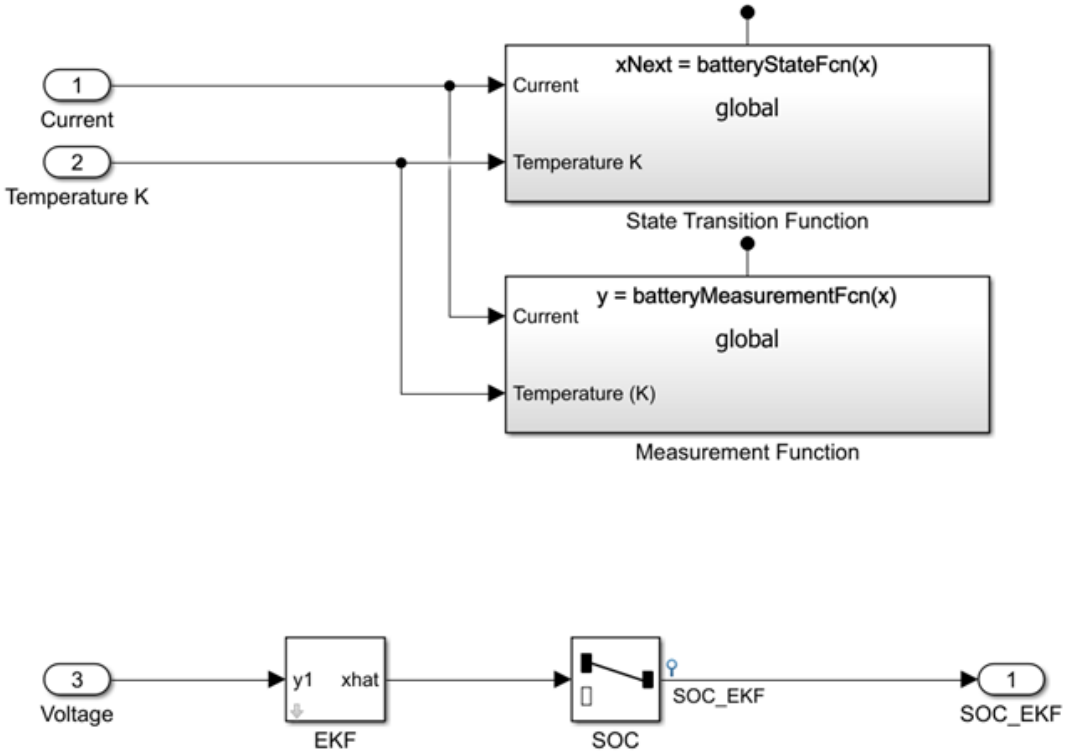

FIGURE 3.6: Extended Kalman Filter Estimator

3.3 Balancing Logic

The Balancing Logic section is one of the most crucial feature of a BMS. In order to guarantee the highest capacity of the battery pack, it is essential to keep as low as possible the voltage gap between cells connected in series. Particularly, the discharge of the battery pack is bounded by the lowest voltage cell, while the charge is limited by the highest one. For these reasons, the voltage gap target is the most important parameter of the balancing logic and it can be changed in the system setup file.

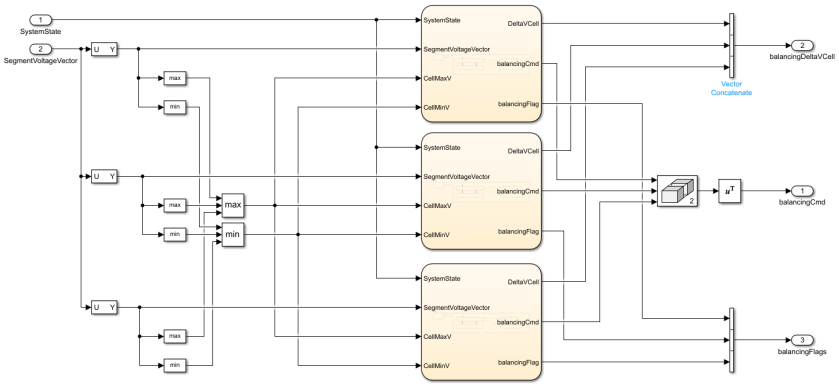

FIGURE 3.7: Balancing logic subsystem.

As it can be seen in Fig. 3.7, the required input signals are:

- **SystemState**: SystemState_t class enum used to activate balancing logic only when Balancing state is selected;

- **SegmentVoltageVector**: [3x10] double array containing cell voltages.

The whole logic is based on the FSM showed in Figure 3.8 that will be briefly described below.

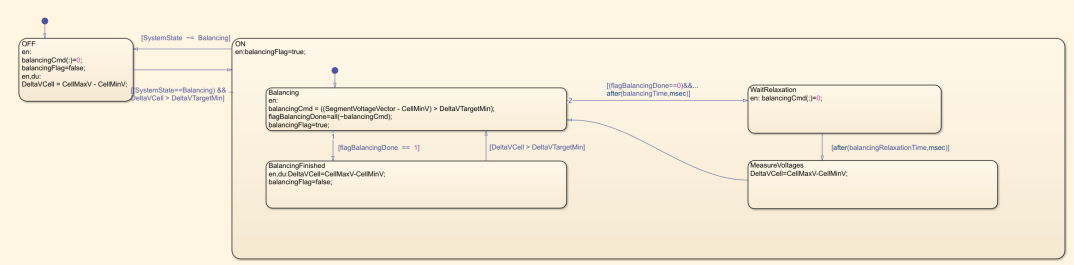

FIGURE 3.8: Balancing logic Stateflow chart.

The Balancing system remain in OFF state until SystemState changes to Balancing and the worst case voltage gap between cells is higher than the DeltaVTargetMin parameter described before. During the OFF state, balancingCmd array is equal to 0, in order to impose null discharge current through the discharge resistor described in Subsection 2.2.1, and the balancingFlag is false.
As soon as, SystemState changes to Balancing and the DeltaVTargetMin condition is not fulfilled, the balancing system starts executing the balancing routine. This routine is based on three different task:

- **Balancing**: during this state the balancingCmd are calculated, based on the SegmentVoltageVectore and CellMinv. For example, if the voltage of cell i-th from segment j-th, minus the voltage of the lowest SOC cell is higher than DeltaVTargetMin means that cells must be discharged activating the corresponding MOSFET, imposing balancingCmd(i,j)=1. If balancingCmd is equal to 0, then the system does not need to be balanced anymore and the Balancing state change to BalancingFinished. Otherwise, if the system requires balancing, after a configurable balancingTime, the balancing state change to WaitRelaxation;

- **WaitRelaxation**: to perform a correct measure of the cell voltages, it is important to measure the Open Circuit Voltage (OCV) between cells. In order to do this, during the WaitRelaxation state, balancingCmd is equal to 0 in order to switch off all the MOSFETs in the discharge circuit and allowing cell voltage to reach their OCV after a configurable balancingRelaxationTime;

- **MeasureVoltages**: at this stage, the relaxation behaviour of cells elapsed and voltages can be measured. After that the routine starts again from Balancing state.

3.4 Fault Monitor

The Fault Monitor logic, depicted in Figure 3.9, is the high level monitor of the whole system state. Particularly, it is in charge of monitoring the safety critical signals of the system and trigger a warning or fault when they are out of their correctly behaviour scope. Each signal is monitored through two parameters that are configurable through the system setup file, and are used to compare the input signal with a threshold value. The warning threshold is a "low weight" fault, that can be used,

for example, to trigger an alarm LED to the user or to show a popup message in the UI. The fault threshold instead is thought to trigger a certain procedure when one or more signals reach their critical value. This kind of signals can be used for example to stop the power delivery of the battery pack to the inverter/charger.

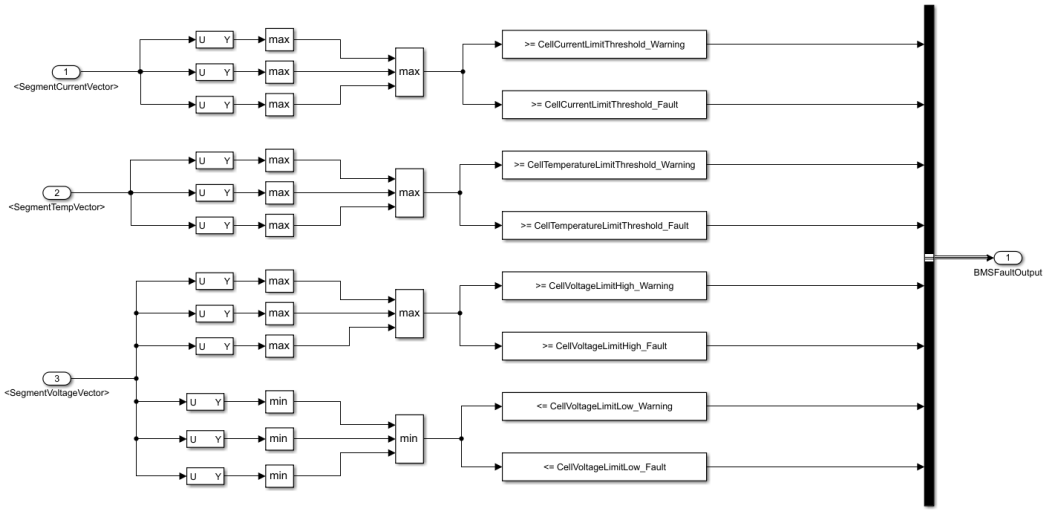

FIGURE 3.9: Fault Monitor subsystem.

As it can be seen in Figure 3.9, the fault conditions that have been implemented are:

- Over Current;
- Over Temperature;
- Under Voltage;
- Over Voltage.

Chapter 4

Testing

In this Chapter the tests performed on the developed controller will be discussed. This will involve both the verification of the correctness of the software behaviour as well as the validation of all the different functionalities and the real-time execution constraints.
The performed simulations are:

- Model-in-the-loop (MIL);
- Software-in-the-loop (SIL);
- Processor-in-the-loop (PIL).

The first one has as main goal the validation of the controller functionality and will be analysed in the next section. The SIL test instead, is an intermediate step between MIL and PIL, used to verify the functionalities of the compiled Controller through the Emebedded Coder. The third one, discussed in Section 4.2, is used to verify that the used processor board is capable of running the developed controller code. Before starting with the description of the performed test, it is worth noting that for both the MIL and SIL/PIL verification phase, a common **Test Sequence** has been used in order to compare and analyse the results obtained, and it is reported in Table 4.1.

Step	Transition	Next Step
Init	after(500, msec)	DrivingPrecharge
DrivingPrecharge	DriveEn == 1	Driving10A
Driving10A	after(40000,msec)	Driving20A
Driving20A	after(30000,msec)	Driving25A
Driving25A	after(20000,msec)	Parking1
Parking1	DriveEn == 0 && DisInverter == 0	ChargingPrecharge
ChargingPrecharge	ChargeEn == 1	Charging2A
Charging2A	after(5000,msec)	Parking2
Parking2	ChargEn == 0 && DisCharger == 0	FaultInjection
FaultInjection	after(200,msec)	RemoveFault
RemoveFault		Balancing
Balancing	balancingFlags == 0	ParkingEnd
ParkingEnd		

TABLE 4.1: Test Sequence used for MIL and PIL.

A brief description of the different steps is now reported:

- **Init**: this step performs the initialization procedures for the system, in particular:

  - System state set to Parking;
  - Driving and charging current set to zero;
  - All fault disabled.

  After 500 milliseconds, it passes to DrivingPrecharge.

- **DrivingPrecharge**: during this step the SystemState is set to Driving and then the step sequence waits the DriveEn signal to be equal to 1, implying that the precharge phase for the inverter contactors has been completed and the system can start driving, passing to the Driving10A step.
- **Driving10A** : in this state a driving current of 10A is imposed for 40 seconds, then it proceeds with Driving20A.
- **Driving20A** : in this state a driving current of 20A is imposed for 30 seconds, then it proceeds with Driving25A.
- **Driving25A** : in this state a driving current of 25A is imposed for 20 seconds, then it proceeds with Parking1.
- **Parking1** : during this step it is tested the inverter discharge phase. SystemState is set to Parking and the system waits that the discharge operation is completed (DrivetrainEn equal to 0 and DischargingInverter equal to 0) to proceed with ChargingPrecharge.
- **ChargingPrecharge** : during this step the SystemState is set to Charging and then the step sequence waits the ChargeEn signal to be equal to 1, that implies that the precharge phase for the charger contactors has been completed and the system can start charging, passing to Charging2A step.
- **Charging2A** : in this state a charging current of 2A is imposed for 5 seconds, then the system proceed with Parking2.
- **Parking2** : during this step the charger discharge phase is validated. System state is set to Parking and the system waits that the discharge operation is completed (ChargEn equal to 0 and DischargingCharger equal to 0) to proceed with FaultInjection step.
- **FaultInjection** : during this step, the system injects a short circuit fault in the first cell of the battery pack for 200 mseconds. This step is required to verify the reaction of the system to this kind of fault and also to create a voltage gap from the faulted cell in order to allow the system to perform the balancing operation. After this, the system proceeds to the RemoveFault step.
- **RemoveFault** : in this step the previous injected fault is removed and the system proceeds to the Balancing step.
- **Balancing** : during this step the system balancing operation is tested. The system will wait until all the segments finish the balancing operation setting the relative balancingFlag to off. When the balancing operation is completed, the system proceeds to the ParkingEnd step;
- **ParkingEnd** : this is the final step of the sequence. SystemState is set again to parking and all the other parameters are set to their initial values.

4.1 Model-in-the-loop (MIL)

In this section the MIL verification phase is described. As we know, Model-in-the-loop test is the first step, after development, that must be performed when dealing with Model-Based Design approach. This kind of simulation has the aim of verifying the correct behaviour for both the Plant and the Controller developed.
In order to succeed with this goal a User Interface has been developed. The latter is reported in Figure 4.1 and it is based on buttons, sliders and rotary switches to help the user in generating Stimulus for the system. In addition, in order to verify the behaviour of the system, Lamp indicators and scopes have been used.

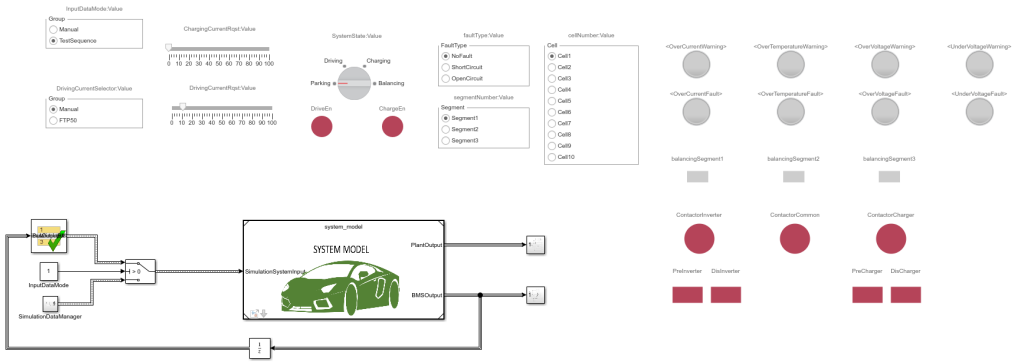

FIGURE 4.1: MIL user interface.

As it can be seen from Figure 4.1, the input Stimulus that can be applied to the system are:

- **InputDataMode** : through this radio button it is possible to switch between testing the system manually or using the Test Sequence described before.
- **DrivingCurrentSelector** : this radio button can be used to select how the driving current request from the user can be requested to the system. Choosing the manual option, the driving current will have a value that can be imposed from the DrivingCurrentRqst slider, instead in the FTP50 option, the driving current will follow the FTP50 emission test cycle standard (Figure 4.2).

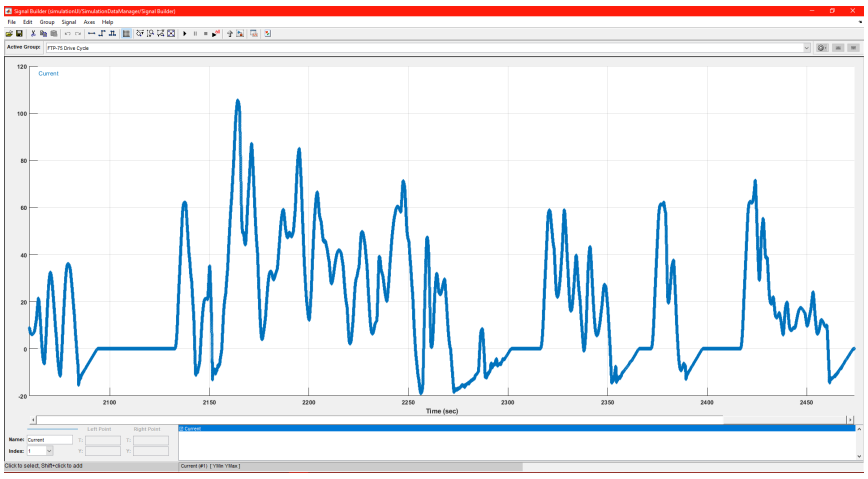

FIGURE 4.2: FTP50 current profile.

- **DrivingCurrentRqst** : slider used to manually select the amount of driving current required from the battery pack;
- **SystemState** : this rotary switch can be used to select one from the four driving mode:
  - *Parking*: in this state the vehicle is parked and everything is in standby mode. This implies that all the contactors of the system are closed and no balancing action is performed;
  - *Driving*: in this state the system performs all the required operation to connect the battery pack with the motor control unit. As a consequence, after that the pre-charge phase is completed, common and inverter contactor will be closed;
  - *Charging*: this state must be selected when the vehicle is connected to a charger station in order to perform all the actions required to connect the battery pack with the battery charger;
  - *Balancing*: during this state, the vehicle starts the balancing operation. This state can be selected every time that a balancing operation is required. When the voltage gap between cells reaches the DeltaVTarget parameter, the balancingSegment flag will turn off.
- **FaultType**, **segmentNumber**, **cellNumber**: as already described in Section 2.2.1, the cell model has been designed in order to be able to inject a Short or Open circuit fault to the plant and test the Controller response to this kind of event. From the FaultType radio button, it is possible to select which kind of fault has to be injected to the system, while with segmentNumber and cellNumber the user can select on which cell the fault will occur.

4.1.1 Simulation of MIL

As shown in Fig. 4.1 it is possible to simulate the behaviour of the overall plant in two different ways:

- By manually tuning the knob that determines the state of the plant, the selector bar for currents and injecting faults to single cells;
- By setting a simulation pattern that reproduces the manual tuning in a controlled and repeatable way.

This last approach was chosen for the description of a single simulation run that is reported in this subsection.

Simulation pattern

First thing to be defined is a pattern that emulates the manipulation of the controllable parameters. This change of parameters is defined in time domain in order to have a significant overview of the expected behaviour of the system as it can be seen in Table 4.1. One remarkable advantage of this approach is the repeatability of a particular experiment changing and re-adapting parts of the controller. This was particularly useful in this application to tune some of the parameters in order to achieve better performances.

4.1.2 Precharge and Discharge phase

In this subsection the precharge and discharge phases for both inverter and battery charger are validated. In Figure 4.3 the inverter precharge operation is reported.

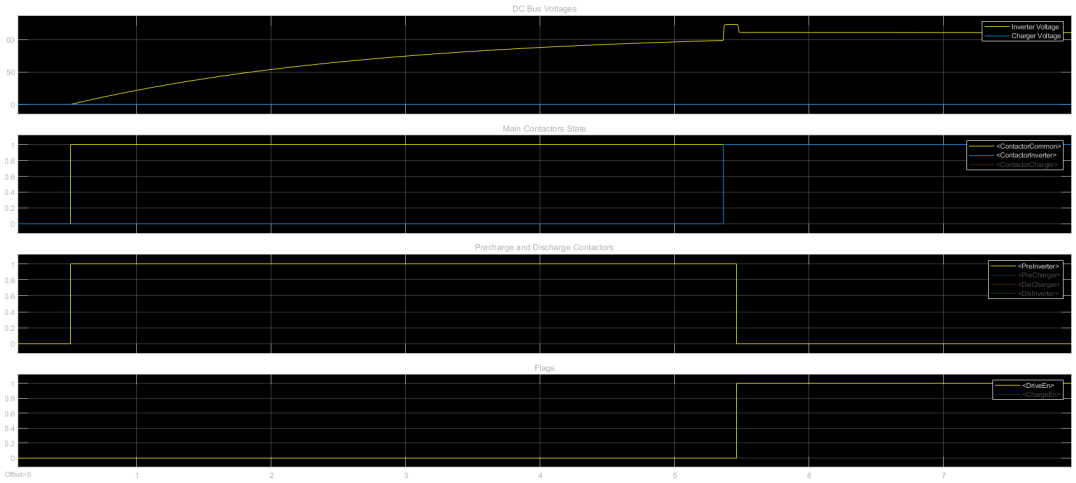

FIGURE 4.3: Inverter precharge operation

As it can be seen from the figure, at time 0.5 s, System State is set to Driving. As expected, the common contactor and the inverter precharge contactor turn on and the inverter voltage on the DC Bus starts to grow. When the inverter voltage reaches the threshold value for precharge operation, the inverter contactor turns on, the precharge inverter contactor turns off and the drive enable flags pass to true, meaning that the precharge operation phase is completed and the system is ready to drive. The same considerations can be extended to the precharge phase for the battery charger, depicted in Figure 4.4.

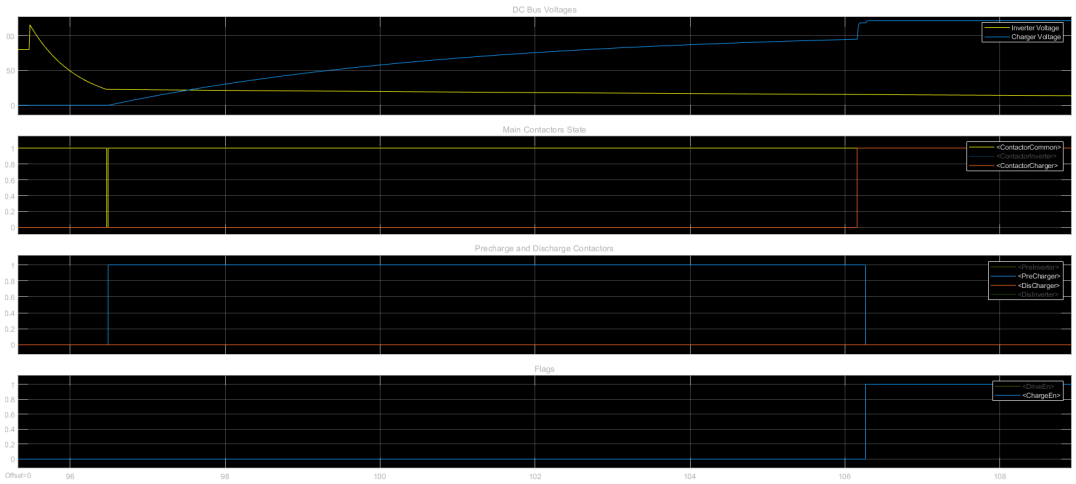

FIGURE 4.4: Battery charger precharge operation

The discharge operation for the inverter instead is reported in Figure 4.5. In this case System State switches from Driving to Parking. When this occurs, the discharge inverter contactor turns on and the contactor inverter immediately switches off and consequently, also drive enable flags becomes false.

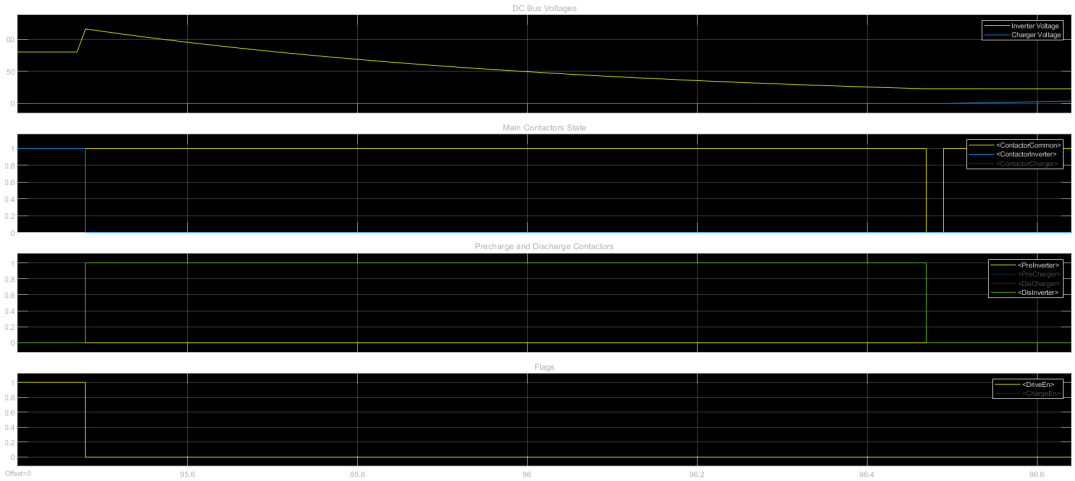

FIGURE 4.5: Inverter discharge operation

The DC Bus starts discharging, until the inverter voltage reaches the lower discharge threshold, meaning that discharge operation is completed. At this stage, the discharge contactor and the common contactor turns off, opening their contact. Also in this case, the same considerations can be extended for the battery charge discharge phase shown in Figure 4.6.

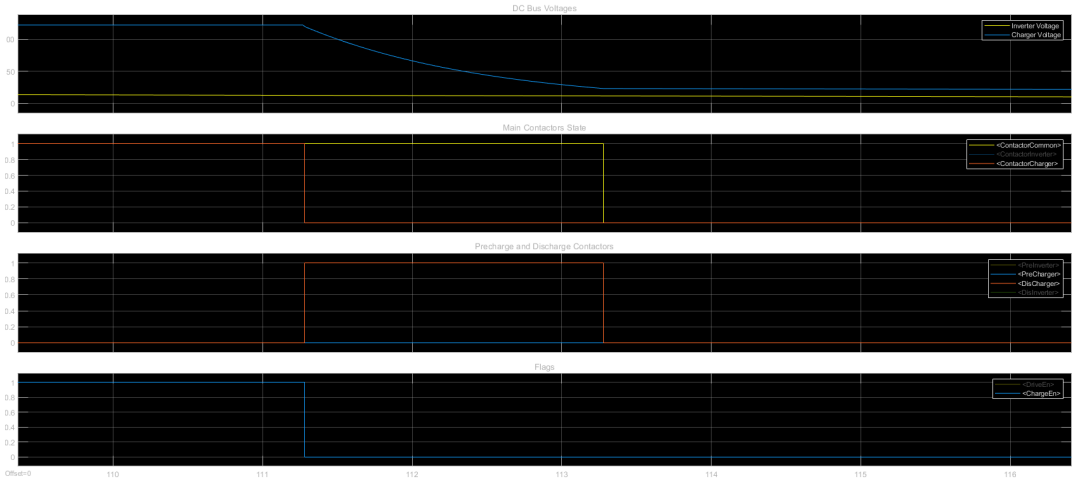

FIGURE 4.6: Battery charger discharge operation

Fault Injection and Cell Balancing

In Fig. 4.7 is shown a condition where cells are firstly in a balanced condition.

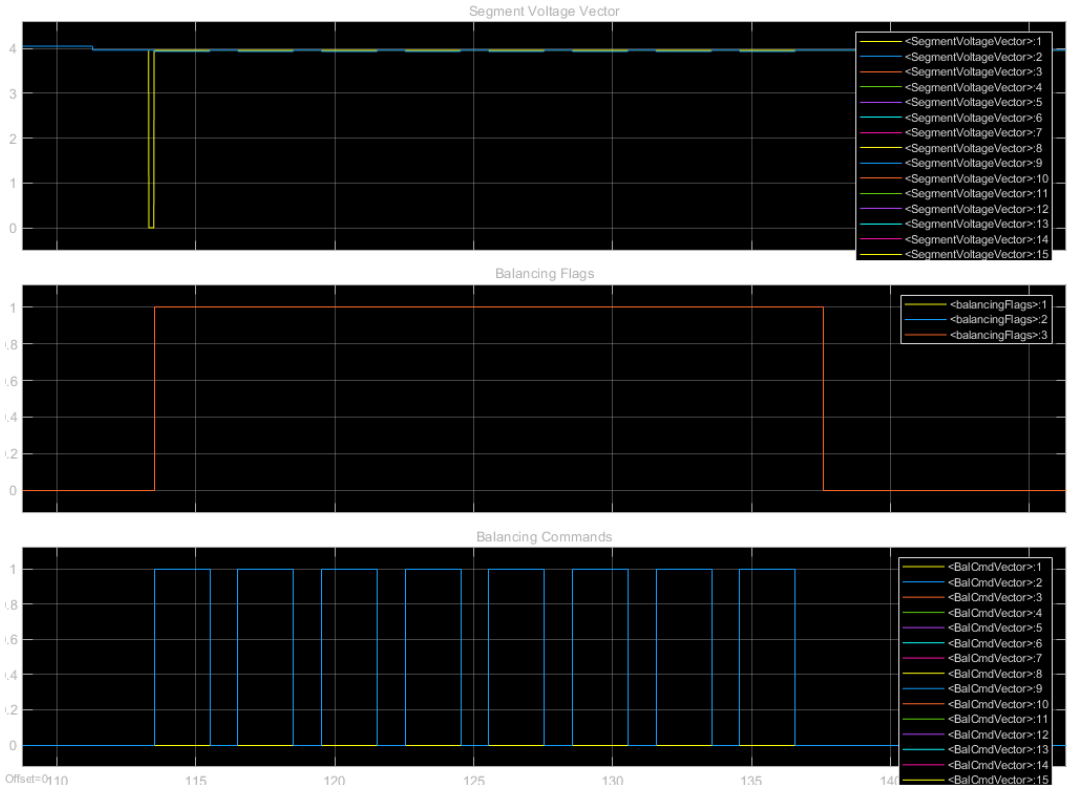

FIGURE 4.7: Fault Injection followed by Cell Balancing

At a certain time, a Short-Circuit fault is injected in the controller and as a consequence cells are not balanced anymore. The top picture of Fig. 4.7 refers in fact to the voltages of the cells. As it can be seen, the yellow signal referring to a particular cell reveals a 0 voltage for a certain amount of time. In response to this the controller suddenly sets the balancing flags to a True condition as it can be seen from the orange signal in the central Figure. Bottom figure instead shows the real action of the controller: it enables for a specific amount of time Balancing Commands in order to discharge cells that have a voltage greater than the one of the cell which experienced a Fault condition. This phenomenon can be appreciated in the Voltage screen where, after an initial divergence, voltages referring to cells return to be overlapped and equal.
The injection of this fault, moreover, generates two other faults that are reported on the Simulation user interface. The timing in which those occur in this particular simulation is well-depicted in Fig. 4.8.

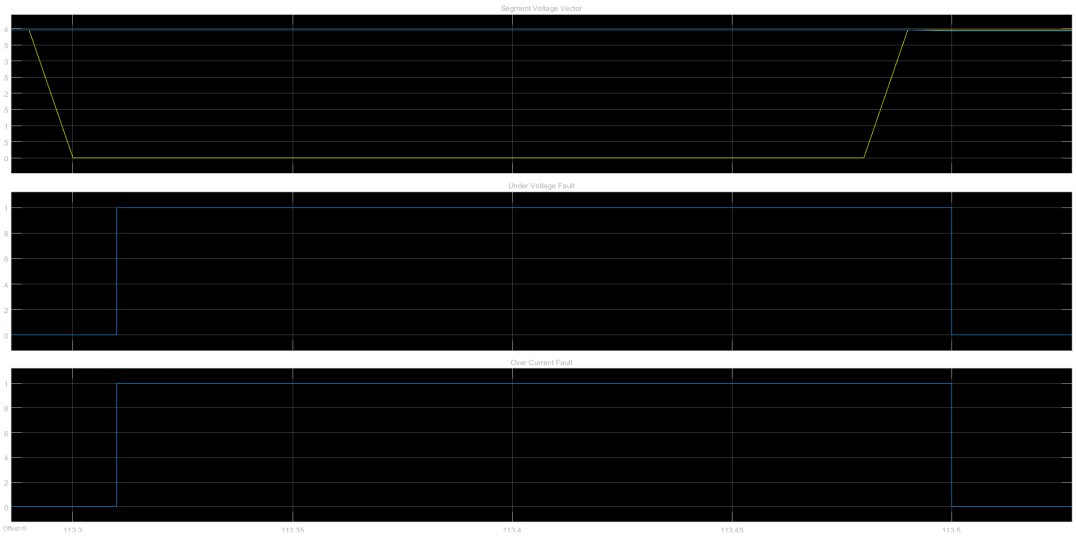

FIGURE 4.8: Faults produced by Short Circuit on a cell

At the top it is possible to spot, like in the previous case, the voltage of the cell that reported the fault compared to the voltage of normal-operating cells. After the faulty cell reaches a 0 voltage condition, the Under Voltage flag is raised to high as well as the Over Current flag.

State of Charge estimation

In Fig. 4.9 is shown a comparison between the real state of charge of a cell (in blue) and the estimated state of charge (in yellow) with the Coulomb Counter method.

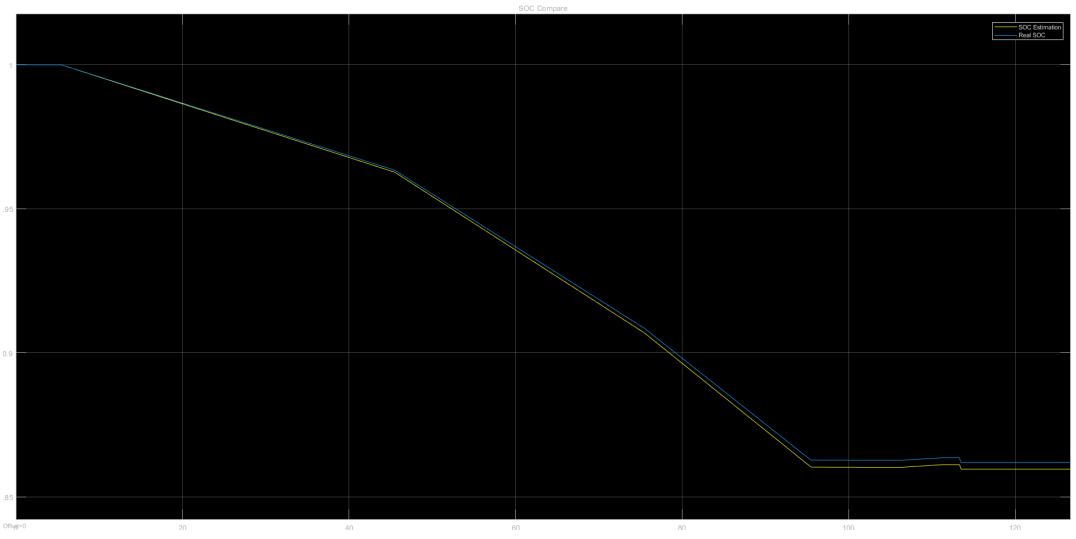

FIGURE 4.9: State of Charge estimation with Coulomb Counter method

The difference between the two signal shows a small divergence just after few moments of simulation, after some actions of the pattern have been performed. This is mainly due to the fact that the relation between discharge rate and cell temperature is simplified in this method by a look-up table that has been manually tuned. As anticipated in Section 3.2, the Coulomb Counter method can not recover from an error of estimation and its integrative term adds constantly an error making the

estimation divergent with respect to the real quantity.
However, the battery pack can be re-calibrated at known points, like for example when it is fully charged or discharged. This is possible if the error performed by the Coulomb Counter is acceptable, if it is not a more sophisticated estimator like Kalman filter can be deployed.

Over-Current warning

Even if the designed simulator can handle a current consumption or injection that raises up to 100 Amperes, a warning flag has been implemented to notify if the current exceeds a threshold of 20 Ampere.

FIGURE 4.10: Warning produced when current injected or absorbed gets over 20A

In Fig. 4.10 is shown how this flag operates: as long as the current remains bounded between 0 and 20 amperes the warning is disabled and when it gets over 20 Ampere it is activated until the condition is not verified anymore.

4.2 Software-in-the-loop (SIL) and Processor-in-the-loop (PIL)

After the validation of the strategies functionalities, through the MIL test, it was possible to proceed in compiling the controller developed and testing it both on the host processor to perform the Software-in-the-loop validation and on a target processor to perform the PIL validation. The model has been compiled using Embedded Coder with the ert.tlc system target file and C Language.

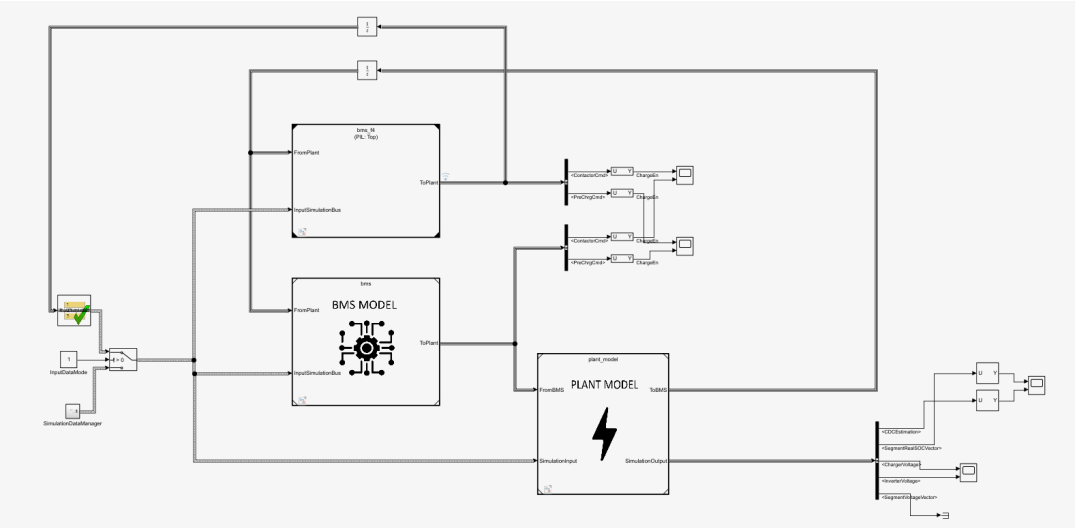

FIGURE 4.11: SIL and PIL simulation interface

Processor-in-the-loop test is the last step that has been made to validate the BMS System. As already mentioned, the aim of the PIL validation is to verify that the Controller developed can be executed on a target processor, satisfying both functionalities and execution timing constraints. To succeed with this goal, the SIL/PIL Manager tool provided by Simulink has been used. This powerful tool, gave us the possibility to compare the execution of the compiled Controller software on both the host (SIL) and target processor (PIL) and analyse any difference in the execution time.
Thanks to the Matlab support for some STM32 Discovery board, this test has been run on an STM32F4-Discovery board, based on the STM32F407G microcontroller. The code has been compiled using Simulink Embedded Coder and the support package for STMicroelectronics Discovery boards.

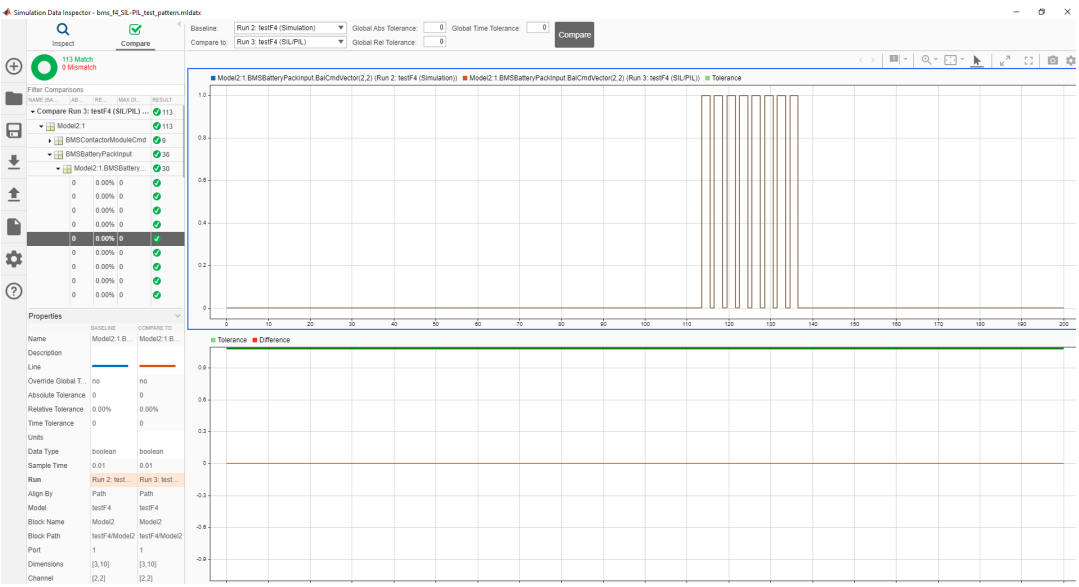

FIGURE 4.12: PIL Inspector: Balancing signals

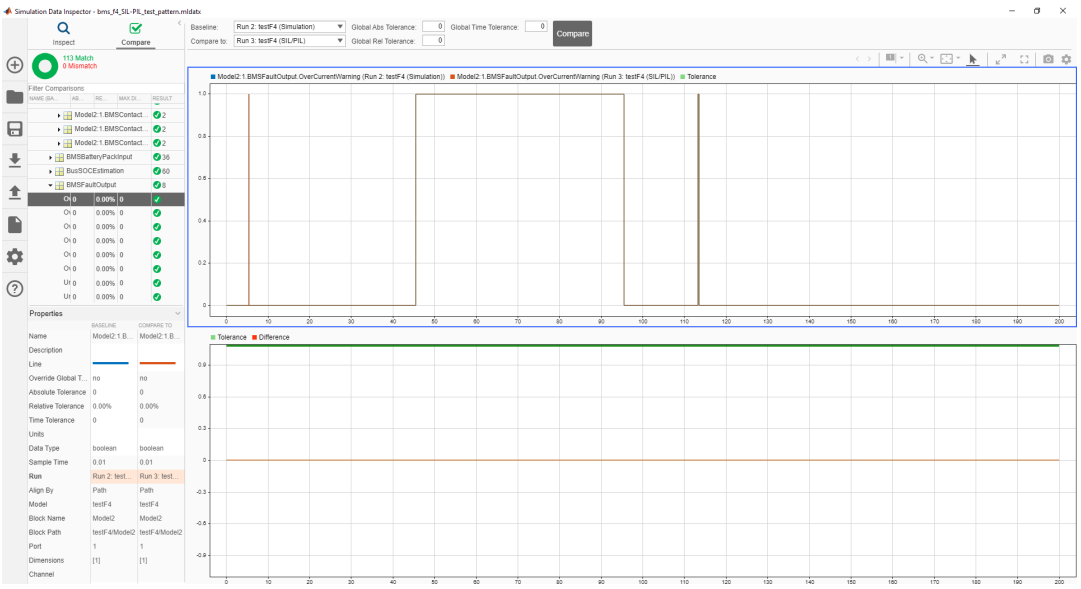

FIGURE 4.13: PIL Inspector: Over current warning

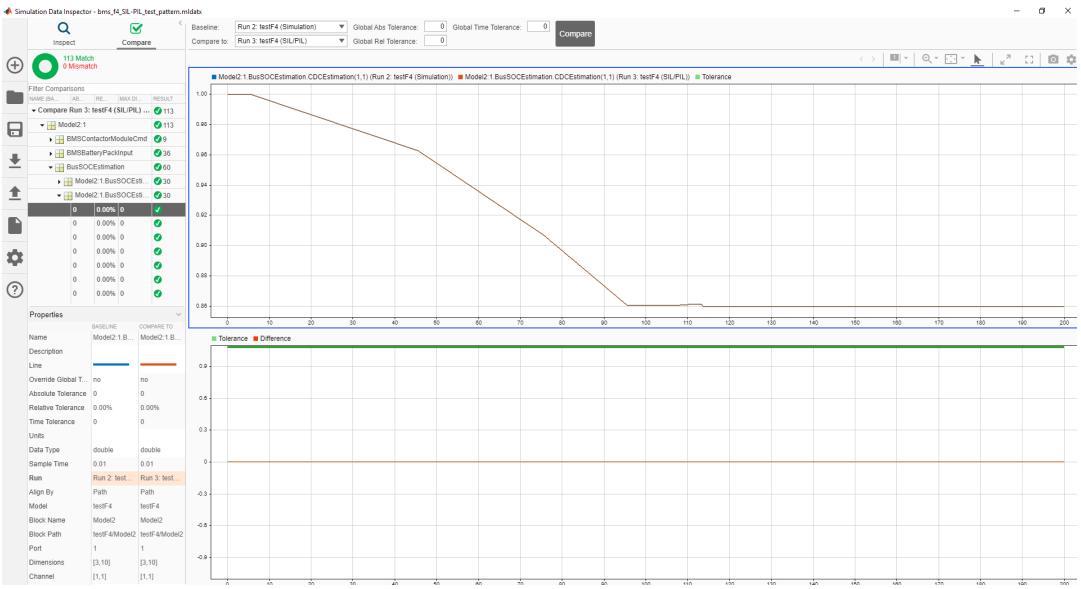

FIGURE 4.14: PIL Inspector: SOC estimation

The results obtained from the SIL/PIL Manager are shown in Figure 4.12, 4.13 and 4.14, representing respectively some details of the Balancing signals, Over current warning and SOC estimation computed from the STM32F4 microcontroller. One of the most relevant results that can be appreciated from the previous graphs, is that all the signals executed on the STM32 Discovery board perfectly match the ones executed on the host processor.
This result validates the fact that the execution of the Controller on a real target processor satisfies the timing constraints designed for the system.
In particular, in Figure 4.15, the execution time for the three main functions generated from Embedded Coder are reported: *bms_f4_initialize*, *bms_f4_step* and *bms_f4_terminate*.

FIGURE 4.15: Profiling Test Report on STM32F4 Discovery

Chapter 5

Conclusions and further improvements

The activity reported in this paper highlights different considerations that is worth expanding.
First of all the approach to the proposed work has been a key point for the accomplishment of the project. Programming and articulating the assignment on different levels allowed to have a clear pattern of the activities that had to be carried out and increased consistently the efficiency and productivity of each team member. The core of the simulations progression was the clear and explicit abstraction that allowed to divide the work on several levels.
The first point leads to different advantages experienced among the testing phases of the designed system. Each one of the designing phases was in fact followed by a severe test phase and, because of the multi-level approach mentioned before, it was significantly more efficient and less time consuming. Even for example going through a generic problem during Processor in the loop simulations, the diagnostic set up along many other tools provided by the simulation environment helped in identifying quickly where the issue was and how to solve it.
The last and most important consideration is that the advantages brought by this rigorously partitioned approach should be considered abstracted from the Battery Management System since they could bring the same advantages if applied to any model-based design.
To conclude, a BMS can be a very complex and sophisticated device. Even if the main functionalities have been designed and tested along this application, here are some further improvements that could be developed in future:

- **Active balancing**;
- **State of Charge Estimation with Extended Kalman Filter**;

Active balancing is a challenging concept that increases the efficiency of the BMS. Instead of discharging batteries that have a higher SOC like in the one implemented in this report, it implements a control logic that transfers power from the most charged cells to the less ones. However this system is quite complicated and just a preliminary study was conducted during this work.
EKF applied to the SOC estimation instead guarantees higher precision since it makes a sensor fusion which takes care of different errors that may affect the quantities that are being used. It is computationally more heavier and difficult to design but it is mandatory for all those applications that need a certain precision for the estimation of the SOC.

Appendix A

Battery Parameters

Cell Custom Parameters	Value	Unit
CellThermalMass	5000	J/°C
CellNumberForSegment	10	
SegmentNumber	3	

TABLE A.1: Cell Custom parameters

Thermal Parameters	Value	Unit
AmbientTemperature	25	°C
CellthermalMass	8e-5	J/K
CellArea	1e-100	m^2
CellConvectiveCoeff	0.1e-5	$W/(m^2K)$
Cell2CellConductiveArea	1e-2	m^2
Cell2CellConductiveThickness	0.1	m
Cell2CellConductiveThermConduct	401	S/m
UpperSegmentConductiveArea	0	m^2
UpperSegmentConductiveThickness	0	m
UpperSegmentConducctiveThermConduct	0	S/m
LowerSegmentConductiveArea	1e.2	m^2
LowerSegmentConductiveThickness	2	m
LowerSegmentConductiveThermConduct	401	S/m
UpperSegment2CaseConductiveArea	0	m^2
UpperSegment2CaseConductiveThickness	0	m
UpperSegment2CaseConductiveThermConduct	0	S/m
LowerSegment2CaseConductiveArea	0	m^2
LowerSegment2CaseConductiveThickness	0	m
LowerSegment2CaseConductiveThermConduct	0	S/m

TABLE A.2: Thermal Parameters

Balancing Circuit Parameters	**Value**	**Unit**
PassiveBalancingRes(1)	5	Ω
PassiveBalancingRes(2)	5	Ω
PassiveBalancingRes(3)	5	Ω
PassiveBalancingRes(4)	5	Ω
PassiveBalancingRes(5)	5	Ω
PassiveBalancingRes(6)	5	Ω
PassiveBalancingRes(7)	5	Ω
PassiveBalancingRes(8)	5	Ω
PassiveBalancingRes(9)	5	Ω
PassiveBalancingRes(10)	5	Ω

TABLE A.3: Balancing Circuit Parameters

Battery Pack Busbar Resistance	**Value**	**Unit**
BusbarRes(1)	0.05	Ω
BusbarRes(2)	0.05	Ω
BusbarRes(3)	0.05	Ω
BusbarRes(4)	0.05	Ω

TABLE A.4: Battery Pack Busbar Resistance

SOC Parameters	**Value**	**Unit**
SOCInit	0.95	
Ts	0.01	
LUTBatteryCharge	2.41 2.39 2.3	C
LutBatteryChargeTemp	278.1499938964844 293.1499938964844 323.1499938964844	°C

TABLE A.5: SOC Parameters

Appendix B

Contactor Parameters

Contactor Resistance	Value	Unit
ContactorChargerRes	0.01	Ω
ContactorInverterRes	0.01	Ω
ContactorCommonRes	0.01	Ω

TABLE B.1: Contactor Resistance Parameters

Relay Resistance	Value	Unit
RelayPreChargerRes	0.01	Ω
RelayPreInverterRes	0.01	Ω
RelayDisChargerRes	0.01	Ω
RelayDisInverterRes	0.01	Ω

TABLE B.2: Relay Resistance Parameters

Pre-Charge Resistances	Value	Unit
PreChargerRes	1000	Ω
PreInverterRes	1000	Ω

TABLE B.3: Pre-Charge Resistances Parameters

Discharge Resistances	Value	Unit
PreChargerRes	250	Ω
PreInverterRes	250	Ω

TABLE B.4: Discharge Resistances Parameters

FSM Constants	Value	Unit
VbattThersholdChrg	0.8	
VbattThersoldDis	0.2	
VbattMin	1	V
DrivetrainEnDelay	0.1	s

TABLE B.5: FSM Constants Parameters

Appendix C

Charger and Drivetrain Parameters

Charger Parameters	Value	Unit
ChargerRes	0	Ω
ChargerPasRes	10000	Ω
ChargerCap	5e-5	F

TABLE C.1: Charger Parameters

Drivetrain Parameters	Value	Unit
InverterRes	0	Ω
InverterParRes	10000	Ω
InverterCap	2.5e-5	F

TABLE C.2: Drivetrain Parameters

Appendix D

Fault Parameters

Temperature Parameters	Value	Unit
CellTemperatureLimitThresholdWarning	50 + 273.15	K
CellTemperatureLimitThresholdFault	65 + 273.15	K

TABLE D.1: Temperature Parameters

Temperature Parameters	Value	Unit
CellCurrentLimitThresholdWarning	20	A
CellCurrentLimitThresholdFault	50	A

TABLE D.2: Current Parameters

Voltage Parameters	Value	Unit
CellVoltageLimitHighWarning	4.21	V
CellVoltageLimitHighFault	4.25	V
CellVoltageLimitLowWarning	2.70	V
CellVoltageLimitLowFault	2.65	V

TABLE D.3: Voltage Parameters