Design rules

# Naming convention

## Basic rules for naming parameters or signal

This document presents the naming convention used for the model. Please note that the rules are set to ease the work on the model, and can be modified or improved.

## Signal and parameter definition:

**Signal**: input/output of a Matlab subsystem or block. Can be sent to the workspace using a “to workspace” block as an array for post processing.

**Parameter**: value imported from the workspace as parameter for a block. Three type: single value, vector (one dimension matrix) or table (two dimension matrix).

## Rules:

* Each name is unique.
* The name is kept along the whole model.
* The names are chosen to avoid confusion.
* A name begin with a tag depending:
  + of the subsystem creating the signal;
  + of the subsystem using the parameter;
  + in case of multiple use of a parameter, the subsystem physically “creating” the parameter is used.
* The names are written in:
  + upper case letter for a signal;
  + lower case letter for a parameter.
* A signal send to the workspace keeps its name (in upper case).

## Additional tips:

* Use keywords: if redundant term appears, keywords and abbreviations can be used. Maintaining a list of keywords and abbreviation and names used can be useful.
* Be careful with the units: in parallel with the names, the units of dimensioned signal or parameter are to be standardized. The SI system is to be favored, but other units can be used for better understanding (rotational speed in RPM for example). A variation of unit can be signaled by a keyword.
* Keep simple name for low level usage: if a signal is used at a low level of the architecture or is sent to the workspace (input / output of a main subsystem for example), it’s better to have short and self-explanatory names. More complex names or names variation can be used at high level of the architecture.
* Be constant: the order of the keyword should be constant, and names for similar items should be built in the same fashion.

# Model conventions

## Code Generation

To simplify the Code Generation of the blocks, one block must have **only one input and output**. Therefore, the model is composed by several blocks which have many inputs and outputs. So we could have used “Mux” and “Demux”. They are transformed into “tab” by the Embedded Code Generation of Matlab. But tabs contain the same type of variables and sometimes the signals (input and output) blocks from the model signals are different type. For example, the Rear Axle block takes as input two command signals, a voltage and a speed. So it would have been difficult to retrieve them when coding.

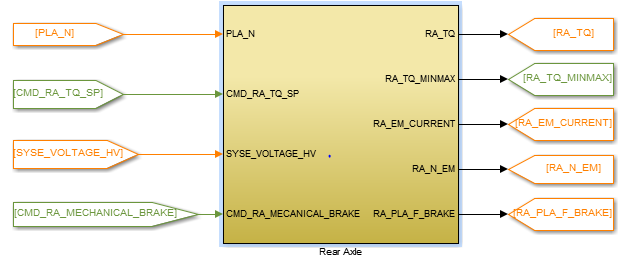


Figure 1: Rear Axle previous version

The solution is to use “Bus creator” and “Bus selector” which is considered as a “structure” which is in Programming language an object which can contain different types of variables.



Figure 2: Rear Axle with buses

Bus Selector

Bus Creator

In the block, in order to simplify the model, we divided the input bus in “goto” signals to realize the functions and then retrieve the outputs with “from” to create the output bus as shown in the following figure.

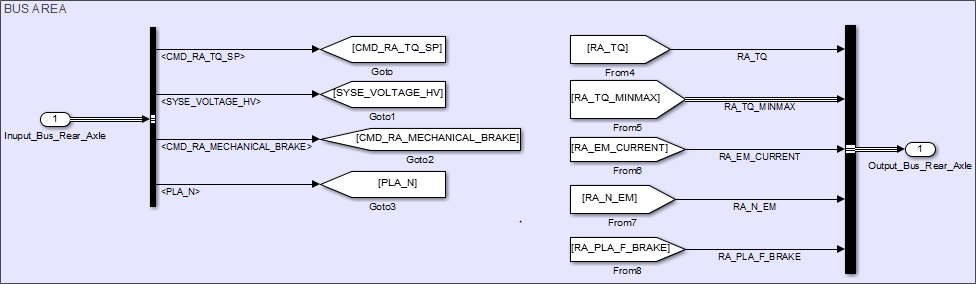


Figure 3: Rear Axle Bus Area

We did the same for every single block that have been auto-coded. The list of all the blocks is in the Sequence Diagram document.

## Color usage

Unless specified, the colors are left basic: black foreground, white background, no drop shadow.

### Subsystem background colors

Two colors are used for distinguishing the subsystems and the functions:

* Green system are Command system; they describe embedded functions
* Yellow systems are physical system; they describe the physical behavior of the component.

Example: the ICE model is separated in two subsystems, a control part representing the ICE ECU functions, and a physical part for its behavior.

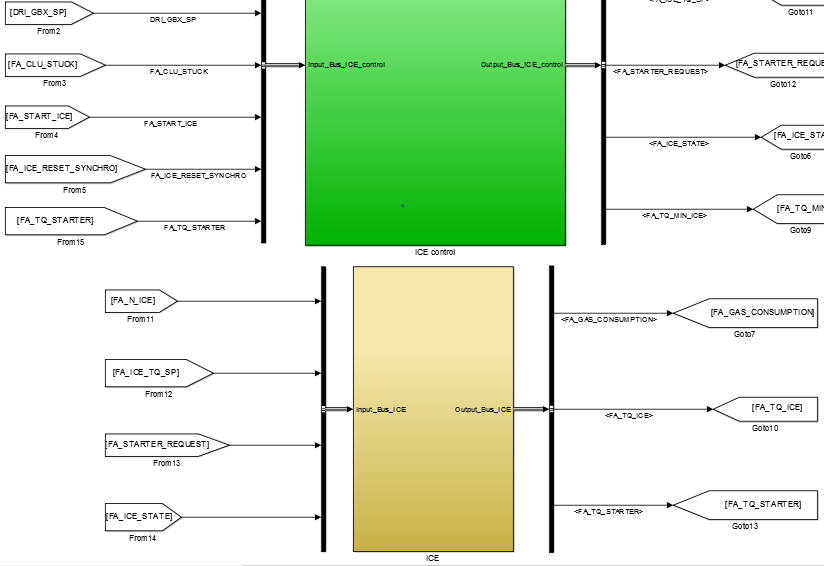


Figure 4: ICE Control and ICE

### The orange boxes

The “delay” blocks with an orange background color have been placed in order to avoid algebraic loop errors and warning. They have no other roles.

Example: a delay has been used in the electrical system on the current loop.

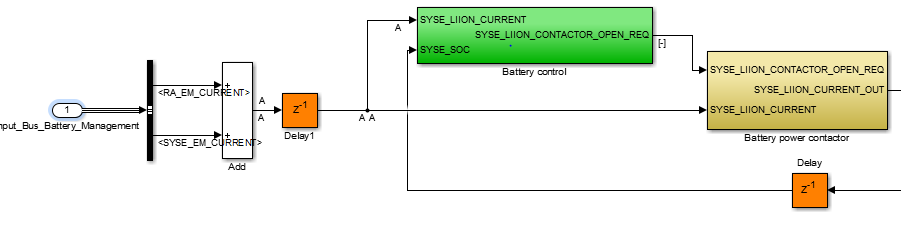


Figure 5: Orange blocks

## Use of “Goto” – “From” blocks

All the “goto” and “from” are local tags. The use of global tag is not recommended.

Local visibility only (option by default)

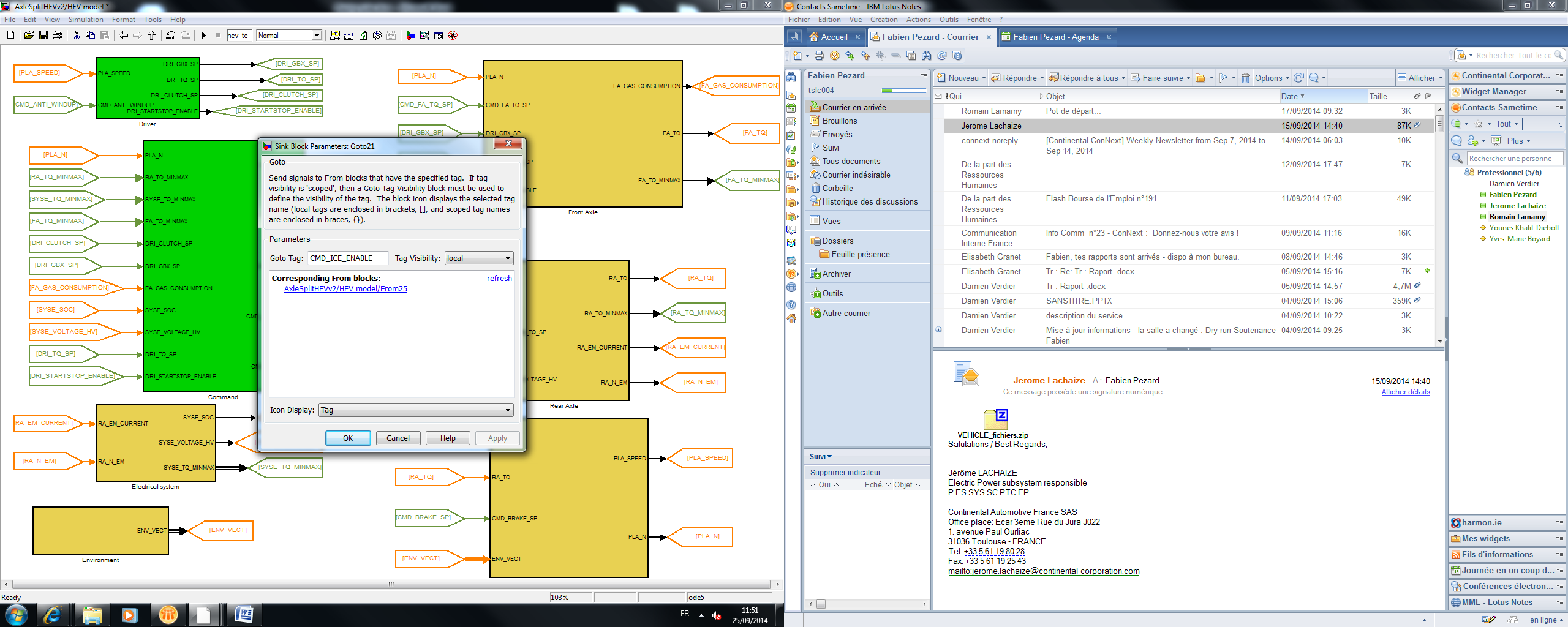


Figure 6: Goto parameters

The use of “Goto” and “from” block is recommended to avoid confusion when too many signals are present.

## Specific case: the first level of the model

On the first level of the model, the use of “goto”-“from” structure is mandatory.

The colors are set on the foreground with the following rules:

* Dark green for command signals
* Orange for physical signals

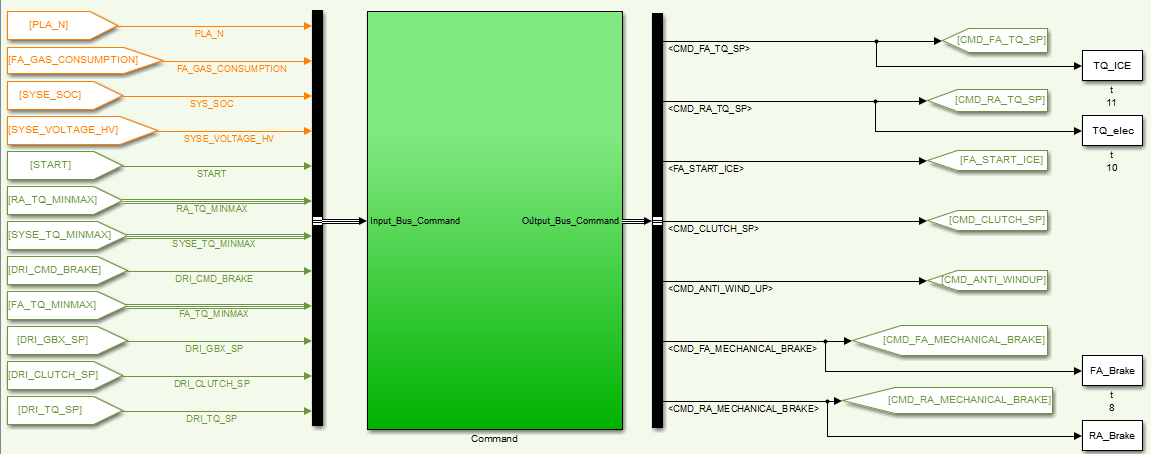
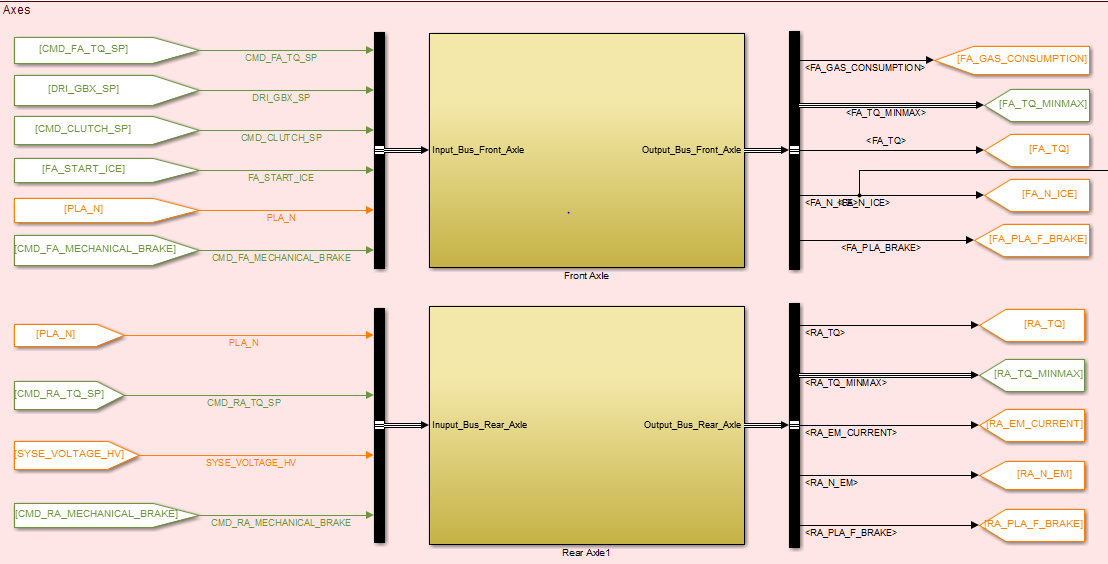


Figure 7: First level of the model

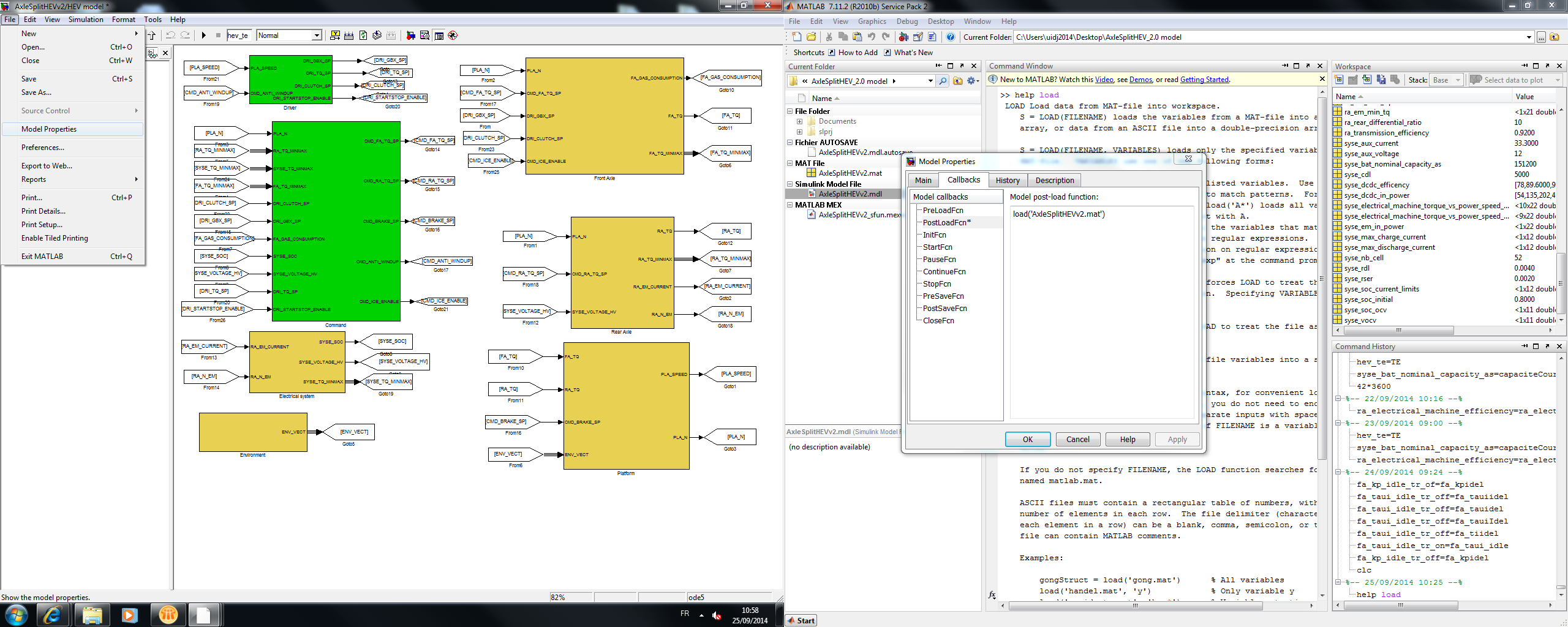
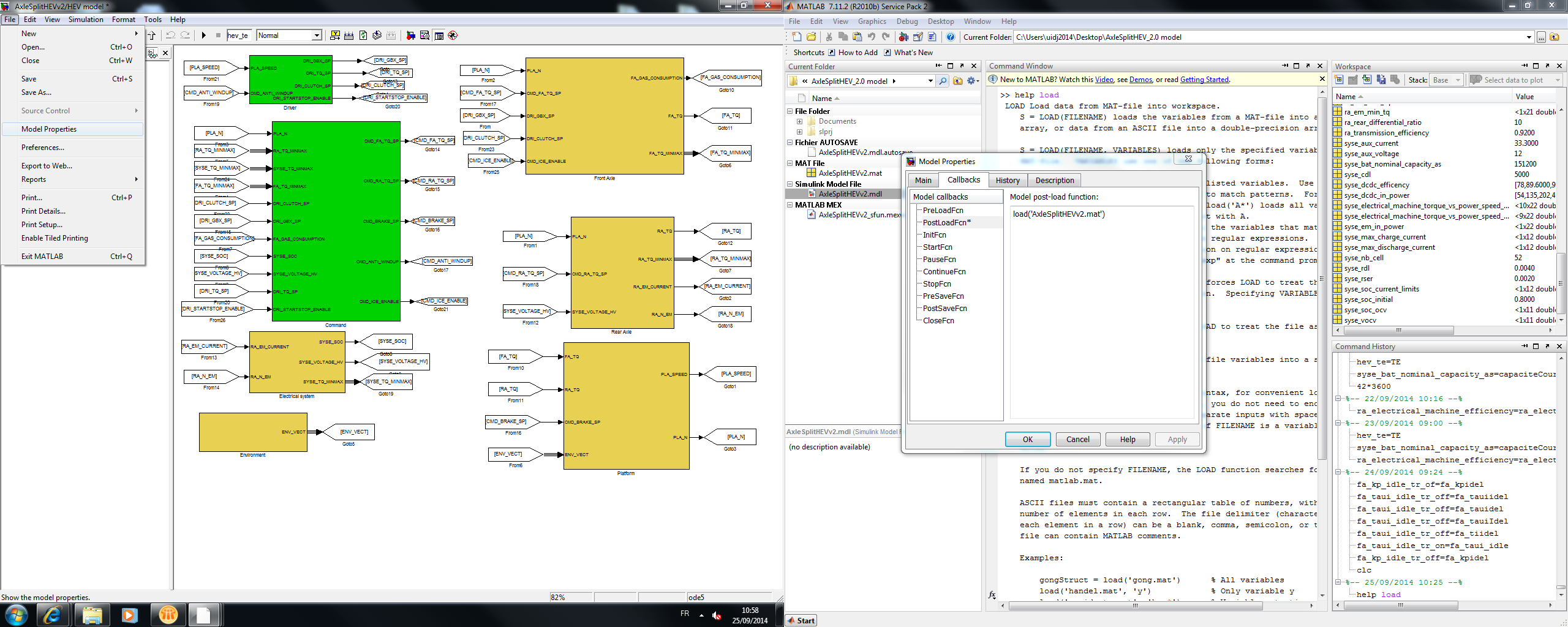
The two command systems (in green) issue command signal, when behavior systems (yellow) issue both type (physical for their behavior part, and command from their low level command part).

## Parameters

The parameters are automatically loaded from the .mat file at the launch. Both file (.mat for the parameter and .mdl for the model) should have the same name.

To modify the automatic load: On the model, click on File🡪Model Properties🡪Data Link Dictionary🡪Callbacks

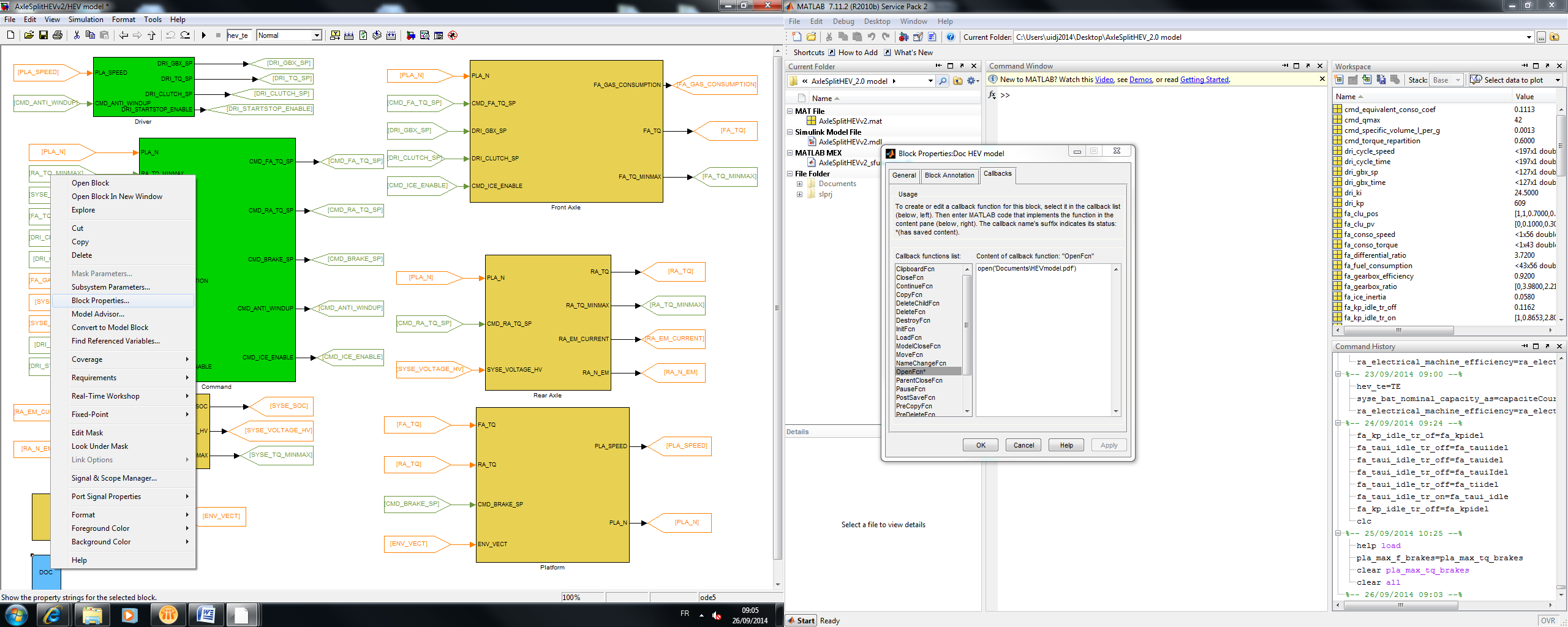
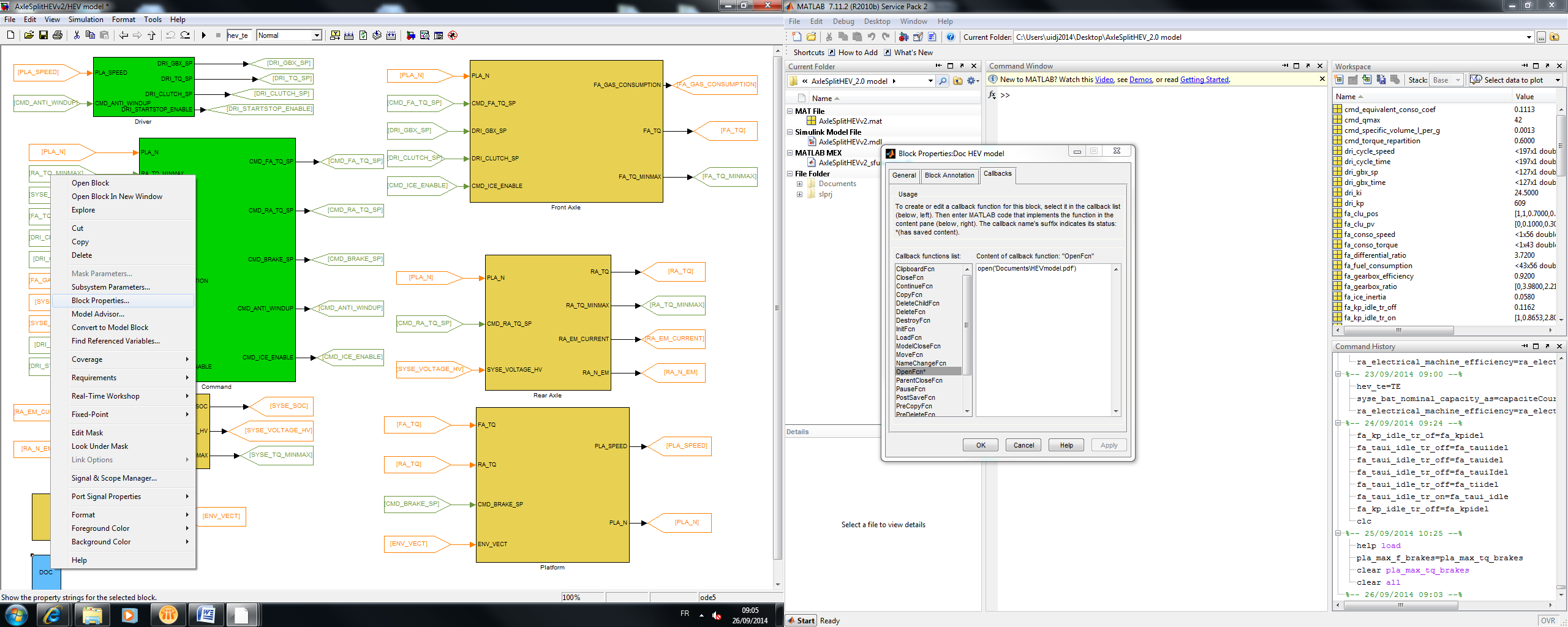
Figure 8: Modification of the parameters



# Document

To create or modify a link to a document:

* Create a empty subsystem;
* On Block Properties🡪Callbacks🡪OpenFcn, add the code: open(‘[filename]’).



In block properties

open(‘[folder]\[filename]’)

The root path of the current folder is implicit, so there are no need to recall the compete path.