International Journal of Innovative Computing, Information and Control
Volume **11**, Number **5**, October **2015**

ICIC International ©2015 ISSN 1349-4198
pp. 1495–1507

MODEL-BASED DESIGN OF EXTERIOR LIGHTING CONTROL FUNCTION FOR AUTOMOBILE: MIL, SIL AND RCP

Max Mauro Dias Santos[1], João Henrique Neme[1], Felipe Rezende Franco[1]
Sergio Luiz Stevan Jr.[1], Weslley Torres[2], Alexandre Baratella Lugli[3]
Armando A. M. Laganá[4] and João Francisco Justo[4]

[1]Department of Electronics
Federal University of Technology – Paraná (UTFPR) – Campus Ponta Grossa
CEP 84.016-210, Ponta Grossa, PR, Brazil
{ maxsantos; sstevanjr }@utfpr.edu.br; neme@outlook.com; felipefranco@alunos.utfpr.edu.br

[2]Department of Automotive Electronics
FATEC Santo André
CEP 09.020-130, Santo André, Brazil
weslley.torres@fatec.sp.gov.br

[3]Department of Industrial Automation
National Institute of Telecommunications (INATEL)
CEP 37.540-000, Santa Rita do Sapucaí, Brazil
baratella@inatel.br

[4]Escola Politécnica
Universidade de São Paulo
CP 61548, CEP 05424-970, São Paulo, SP, Brazil
lagana@lsi.usp.br; jjusto@lme.usp.br

Received September 2014; revised March 2015

Abstract. *Model-Based Development (MBD) is a design methodology, which has been widely used in automotive embedded software. Within this methodology, engineering processes involve both manufacturers and suppliers, which is suitable for the next generation of automotive E/E systems. All players benefit from standardized information exchange, workflow and toolchain. We present a model-based functional safety exterior lighting system, which is an important vehicular function hosted in the electronic control module. This function was developed and tested in Model-In-the-Loop, Software-In-the-Loop and rapid control prototyping development stages, from the supplier side. This demonstrated that, in preliminary and virtual phases, it is important to be able to find bugs and generate quality software when deploying on target systems. The approach considered the software tool to verify the function with a set of test cases that validate the preliminary requirements.*

Keywords: Model-based design, Functional safety, Exterior lighting, Automotive software, Embedded system

1. **Introduction.** Currently, automotive E/E architectures are characterized by a high number of interconnected ECU (Electronic Control Units), up to 70 in luxury cars, with highly complex functions, such that the traditional methods for developing and testing of automotive systems cannot cope with the current market time [1]. The development process of automotive embedded software systems is currently typically a cooperative effort between carmakers and suppliers, changing the role of each player. The development chain includes [2]:

- OEM (Original Equipment Manufacturer): They are the car manufacturers, such as *GM*, *FIAT*, *BMW*, *Ford*, *DaimlerChrysler*, and *Toyota*, that introduce the final products to consumers;
- Tier 1: They are suppliers that provide subsystems, such as power train management, suspension control, and brake-by-wire devices, to OEMs. Examples of suppliers are *Bosch*, *ADVICS*, *Continental*, *Delphi* and *Visteon*;
- Tier 2: They are suppliers of microelectronic and real-time operating systems, such as *ARM*, *WindRiver* and *ETAS*, that provide both OEM and Tier 1; and chip manufacturers such as *NXP*, *Freescale*, *Infineon*, *ST*, and *Renesas*;
- Manufacturer: They are suppliers, such as Flextronics and TSMC, which provide board or partial product assembly.

The vehicles of the next generation will demand a growing number of complex functionalities, and embedded software development becomes one of the bottlenecks to accomplish the process of development, production, services, and time-to-market.

MBD is a design methodology widely used for automotive embedded software, where the engineering process gets together OEM and suppliers, benefiting with information exchange, development workflow and toolchain in a standardized way. Model-based design provides an efficient approach to establish a common framework for communication throughout the design process. A recent investigation examined the costs and benefits of MBD of embedded systems in car industry [3]. For example, in a specific application, MBD has been used for control design of ACC (Adaptive Cruise Control) system [4].

For the functional safety, features form an integral part on development of every automotive product, ranging from specification and design to implementation, integration, verification, validation, and product release. The ISO 26262 is the Functional Safety for Automotive Electric/Electronic Systems, defining the standard for automotive electronic and electrical safety-related systems, applicable throughout their lifecycle [5].

Here, the target system, selected to apply this methodology, is the automotive Exterior Lighting System (ELS). It is responsible for car lighting and signaling functions, as well as its identification by other drivers and pedestrians. We present a model-based functional safety applied to the ELS, which is considered an important function in vehicle embedded in the electronic control module. This function was developed and tested in Model-In-the-Loop, Software In-the-Loop and rapid control prototyping stages, in the supplier side, demonstrating that, in initial or virtual phases, it is important to detect bugs preliminarily and generate high quality software when deploying it on the target system. This approach considers the software tool for the function, with a set of test cases that validate the preliminary requirements.

This paper is organized as follows: Sections 2 and 3 present respectively a background over model-based design and functional safety; Section 4 describes how to apply the MBD for exterior lighting from requirements to tests, Section 5 shows that the verification of requirements is achieved, Section 6 presents the Rapid Control Prototyping, and finally Section 7 presents a summary.

2. Model-Based Development for Automotive Software.

MBD is a mathematical and visual methodology to design complex systems in different domains, such as in automotive, aerospace, motion control and industrial equipment applications. The model system is in the center of the development process, from requirement development, through design, implementation and testing [3].

MBD provides an efficient approach to establish a common framework for communication throughout the design process, while supporting the development cycle ("V" diagram).

2.1. **Background of MBD.** In model-based design of complex control systems, the development is divided into four main steps: 1) modeling a physical plant, 2) analyzing and synthesizing a controller for the physical plant, 3) simulating the plant and controller, and 4) integrating all these phases by deploying the controller. The model-based design paradigm is significantly different from traditional design methodologies. Rather than using complex structures and extensive software codes, the MBD approach allows designers to define models with advanced functional characteristics, using continuous-time and discrete-time building blocks. Those models, combined with simulation tools, could lead to rapid prototyping, software testing and verification. This improves the testing and verification process, in addition to, in some cases, allow the Hardware-In-the-Loop simulation to perform testing of dynamic effects on the system more quickly and efficiently than within traditional design methodologies. It must be considered that this process involves OEM, Tier-1, and Tier-2.

The main steps in MBD approach are as follows.

1) Modelling of physical plant. It could be data-driven or first principles based, in which the data-driven plant modeling uses some techniques, such as system identification. Within system identification, the plant model is identified by acquiring and processing raw data from a real-world system and choosing a mathematical algorithm, in order to identify a mathematical model. First principles modeling is based on creating a block diagram model that implements known differential-algebraic equations governing the plant dynamics. An example is physical modeling, where a model is created by connecting blocks that represent the physical elements of the real plant.

2) Controller analysis and synthesis. The mathematical model conceived in the previous step is used to identify dynamic characteristics of the plant model, such that the controller can be synthesized based on those characteristics.

3) Simulation. In order to solve the problem, usually multiple computation models should be used during the construction of the dynamic system. During simulation, errors due to the two previous steps can be identified immediately, rather than later in the design phase. During this step, it is possible to capture all the system interation, and test against theoretical model, rendering a robust design. Generally, the code synthetizers are incorporated into simulation environment, such as LabVIEW and Ptolemy II.

4) Deployment. At this point, the code generated in step 2) is running in the real system, similar to simulation, allowing to start debugging the real application.

MBD offers remarkable advantages when compared to a traditional approach.

- It provides a common design environment, which facilitates general communication, data analysis, and system verification between development groups.
- Engineers can identify and fix errors early in system design, when the time and financial impact of modifications are minimized.
- Design reuse, for upgrades and derivative systems with expanded capabilities, is facilitated.

2.2. **The X-In-the-Loop – XIL.** The model based workflow is used to create both the controller and the plant, and then these terms represent how real the controller is: Model-In-the-Loop (MIL), Software-In-the-Loop (SIL), Processor-In-the-Loop (PIL), and Hardware-In-the-Loop (HIL) [4].

This investigation focuses only on MIL, SIL and Rapid Control Prototyping (RCP) as the Hardware-In-the-Loop phases. Therefore, the steps in this phase start with the definition of the requirements, then by modeling the plant.

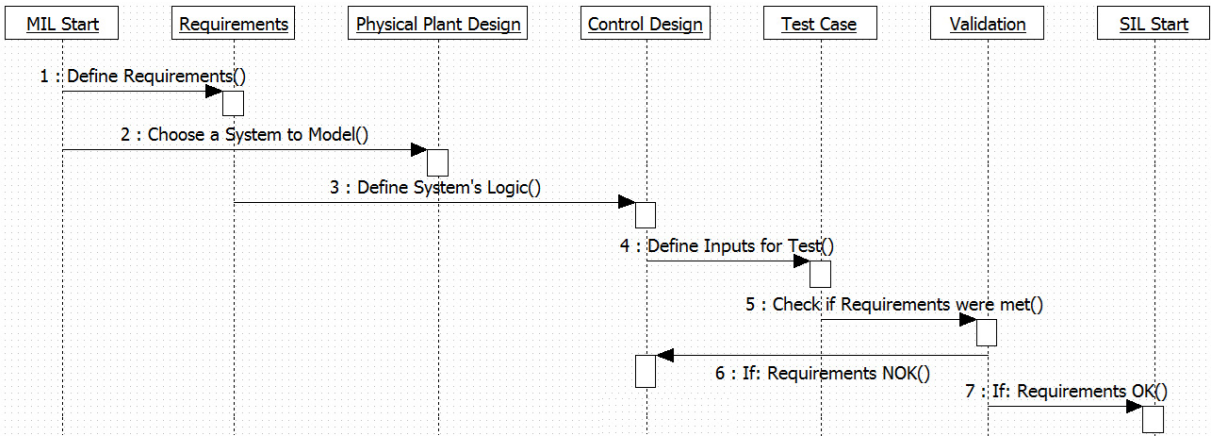

FIGURE 1. Model-In-the-Loop workflow

After modeling the plant, it is possible to define a test case to validate the proposed model with offline simulation via software. Figure 1 represents the MIL workflow. All other phases have the same workflow, using the same logic, inputs and physical plant.

The next step is the Software-In-the-Loop phase, which also allows an offline simulation, although the model is still not verified, but only the C code generated automatically with Simulink® Coder.

The RCP, the last phase of this investigation, is a Computer Aided Control System Design (CACSD). In the RCP, the plant dynamics and/or the controller are implemented in a digital signal processing board, which allows an easy adjustment of various plant and/or the controller parameters [6].

The results of the offline simulation and real time control have shown that the X-In-the Loop can achieve better comfort under assuring stability. They also show that RCP, based on Matlab/Simulink® and dSpace, can shorten the research period of vehicle semi active controller and reduce research costs.

3. **Exterior Lighting Function.** The lighting system is composed of lights and signal devices assembled or integrated into various parts of a vehicle. Those parts may include front, side, rear, and in some cases the upper part of the vehicle. This system allows safe operation of the vehicle at night, increases the visibility for the driver, and facilitates visualization of the vehicle by others.

Exterior lighting system has increasingly become one of the priorities for carmakers. Nowadays, head or tail lamps capture the car signature, and sometimes headlamps are one of the most recognizable elements of a car exterior. Therefore, innovative new lighting designs provide new products that are released on a yearly basis: LED based headlamps, tail lamps, turn indicators, and signal lamps. Here, the functions of ELS which are analyzed are: a) Position Lights; b) Low Beam; c) Full Beam; d) Indicator Lights; e) Reverse Lights, and f) Brake Lights.

3.1. **Requirements.** In order to start the project, the requirements of each subsystem to be controlled must be defined. Requirements are the set of necessary preconditions to activate a lamp, and each manufacturer has its own logic for the external lightning. We have developed generic requirements:

Requirements
To activate Position Lights: • REQ-01: the ignition position must be at "On", "Accessory", or "Crank". • REQ-02: the light switch must be at "Position Lights".
To activate Dipped Beam: • REQ-03: the ignition position must be at "On", "Accessory", or "Crank". • REQ-04: the light switch is at "Dipped Beam".
To activate Full Beam: • REQ-05: the ignition position must be at "On", "Accessory", or "Crank". • REQ-06: the Full Beam Switch must be pulled.
To activate Left Indicator Lights: • REQ-07: the ignition position must be at "On", "Accessory", or "Crank". • REQ-08: the indicator switch must be turned down.
To activate Right Indicator Lights: • REQ-09: the ignition position must be at "On", "Accessory", or "Crank". • REQ-10: the indicator switch must be turned up.
To activate both Indicator Lights: • REQ-11: the hazard-warning switch must be pressed.
To activate Reverse Light: • REQ-12: the ignition position must be at "On", "Accessory", or "Crank". • REQ-13: the reverse gear must be engaged.
To activate Brake Lights: • REQ-14: the brake pedal must be pressed.
For all cases: • REQ-15: the battery tension level must be from 16 to 9 Volts.

3.2. **Functional design.** The behavior of this system was modeled virtually, using a specific computing tool. Using Matlab/Simulink®, combined with the requirements defined previously, a model was designed, such that all tests could be performed.

Figure 2 shows part of the main screen of the ELS system, including Position Lights, Low Beam and Full Beam cases. Other cases are organized below and are not shown in the figure. Each case has its own subsystem, complying with each requirement.

Inside each subsystem, a Stateflow® chart was designed. Each subsystem is fed only with the signals created for its case. The Stateflow® chart used in the Full Beam subsystem is shown in Figure 3. In the figure, the text inside the brackets represents all inputs shown in the left side of the Stateflow® chart. These inputs are listed and explained below:

FBS: Full Beam Switch represents the switch inside the car that activates the Full Beam Lights;

IgPos: Ignition Position represents that it is in one of the positions that can allow all system to work;

FB1: Full Beam Lamp represents a current sensor that can indicate if a specific lamp is on or off. This sensor is used to trigger an error signal if a lamp from an active case is not working.

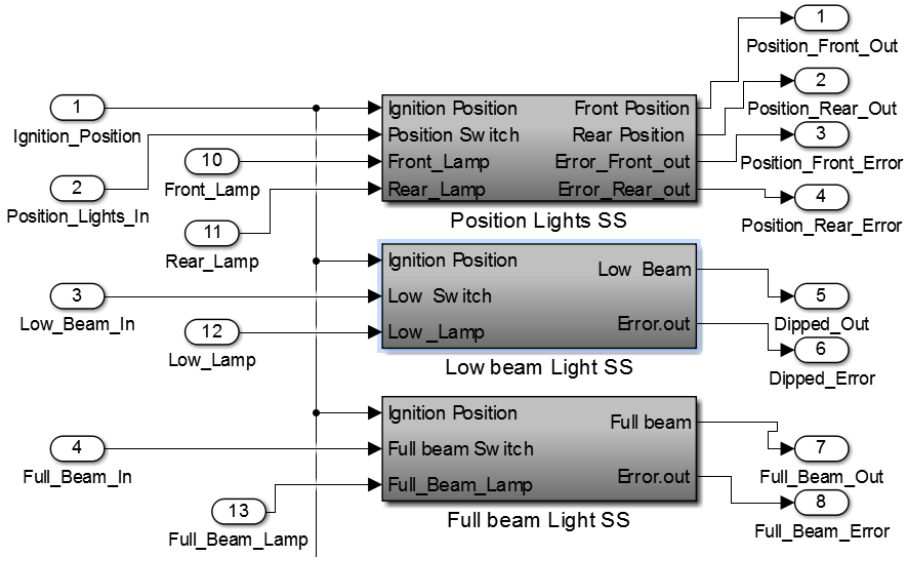

FIGURE 2. ELS model in Simulink®

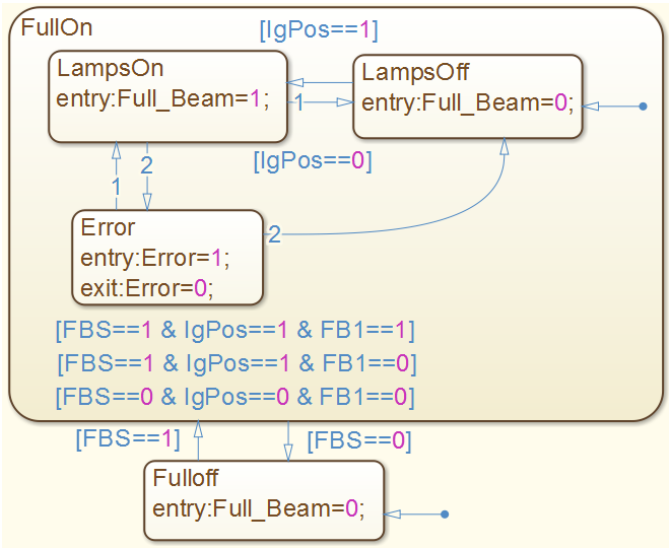

FIGURE 3. Stateflow® chart for the Full Beam subsystem

All cases have similar subsystems, which are distinguished by the respective I/O and Stateflow® chart. It was possible to understand from the Stateflow® charts that the transitions between states are triggered, depending on the situation of each input. In order to activate the full beam light, the Ignition Position and the Full Beam Switch inputs must be high. The logic behind the Stateflow® chart is better illustrated in the block diagram shown in Figure 4.

The next step, after modeling the controller logic, was the simulation phase. In order to perform simulations, some input variables were created and added into the system. Following MBD concepts, the validation was performed in three different phases: Model-In-the-Loop, Software-In-the-Loop and Rapid Control Prototyping.

4. **Model-In-the-Loop – MIL.** In this case, there is a model, built in the Functional Design phase, which performs the controller role to work with a model of the plant. Both models are running within a software. The advantages of this simulation include

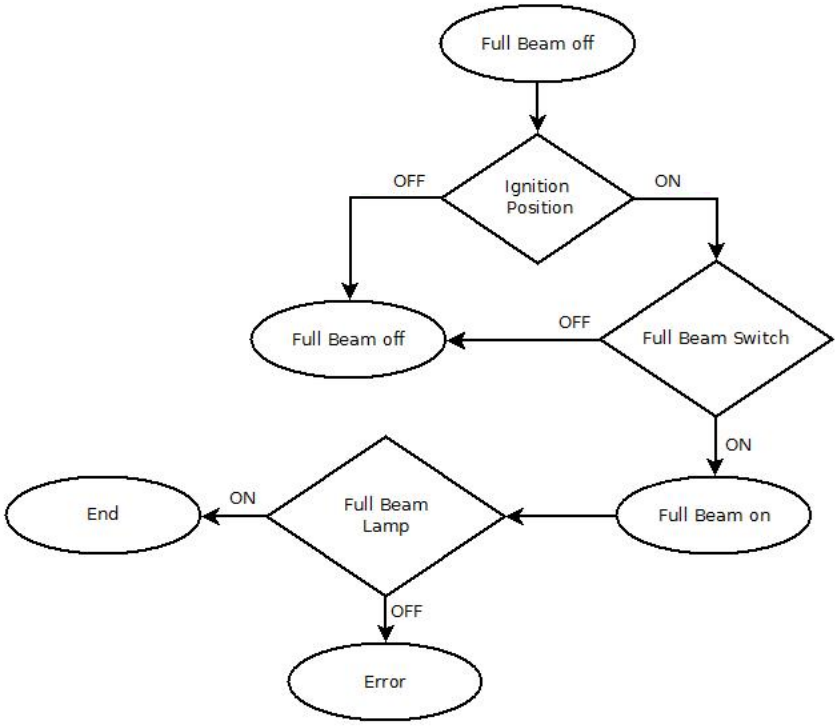

FIGURE 4. Block diagram that illustrates the Full Beam system logic

extreme fast development and the possibility to perform small changes to the controller immediately and test the whole system.

For verification purpose, a case test was formulated with input signals. Each signal was created in order to prove that there were no errors in the logic of the system. Those signals were introduced in the Verification stage.

Other inputs, related to the ignition position and the battery condition, also deserved attention. Therefore, in order to analyze correctly all outputs, it was important to observe the battery level, ignition position, and the hazard warning.

Since the test case was formulated to validate the system, the simulation allowed checking all the outputs and comparing them with their respective inputs. Some inputs, the ignition position and battery level, have a vital role in the system operation. Since both inputs were requirements for the system to function, the system did not work before t = 50 (ignition position) and after t = 760 (battery). The "hazard" warning was responsible for the right and left indicator lights to function simultaneously. Those signals are shown in Figure 5.

Although the time displayed in graphic is 1000s, the real simulation length was 10s. According to the figure, this scenario was designed to prove that there were no errors in the system logic. It is possible to observe that:

- The position light switch is activated before the ignition position is in Acc;
- The hazard-warning button is pressed while the left indicator is active and later when the right indicator is active;
- The battery level reaches less than 9 Volts when several inputs are still active.

Figure 6 shows the resulting outputs for each case, after the simulation, and allows to compare with each input.

It is important to observe the Hazard Warning input, in order to understand the Indicator Lights functions. Those cases were active either when their inputs were activated or

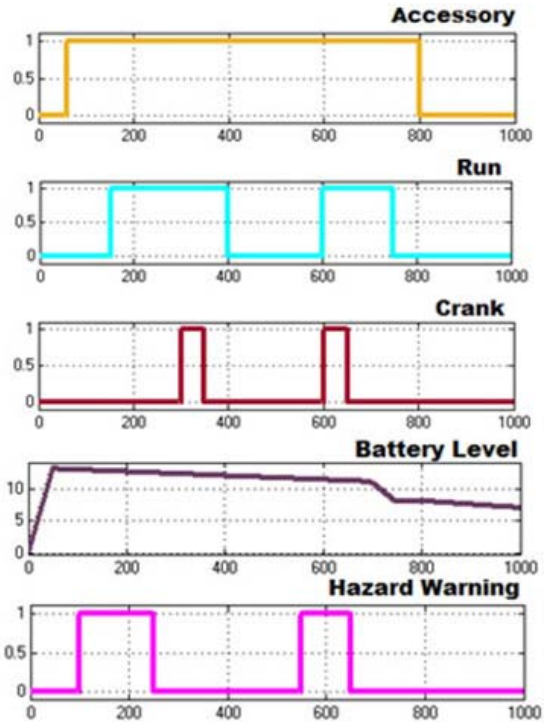

FIGURE 5. Important outputs to prove model logic

when the Hazard Warning was activated. Comparing the outputs illustrated in Figure 6, it was concluded that the requirements were satisfied. The Validation box explains each case separately.

5. **Software-In-the-Loop – SIL.** After the Model-In-the-Loop phase verified the controller against the requirements, it was possible to implement the controller. In the Software-In-the-Loop phase, the controller was no longer running as a Simulink® model. Instead, it was replaced by a software code execution in the simulation, such that this case delivered a more realistic controller, and a C code was tested instead of a system model.

Matlab® was used again to build the code and embed in a Simulink® block, called S-Function. The same Test Case, used in the Model-In-the-Loop phase, was used to verify the controller. Since in Model-In-the-Loop phase, the controller logic was already verified, now to verify with Software-In-the-Loop it was possible to verify its logic by simply comparing the outputs of both phases. Figure 7 shows a comparison between Model-In-the-Loop and Software-In-the-Loop outputs. It is possible to observe that every output behaved as expected, showing that the controller was working properly.

6. **Rapid Control Prototyping – RCP.** With the Rapid Control Prototyping, it was possible to design the control algorithm and perform an offline simulation on a computer. In order to perform RCP, it was necessary to perform some changes on the model developed in Simulink® during the Model-In-the-Loop phase. In order to verify the first model, the outputs were observed with Simulink® scopes, since the process was validated with software simulation. For the RCP, it was necessary to embed the code in an external hardware, the MicroAutoBox® from dSpace®. This required changing the normal outlets for Simulink® blocks of Digital Inputs and Outputs (DIO) of MicroAutoBox®. Figure 8 shows a part of the modified model in Simulink®.

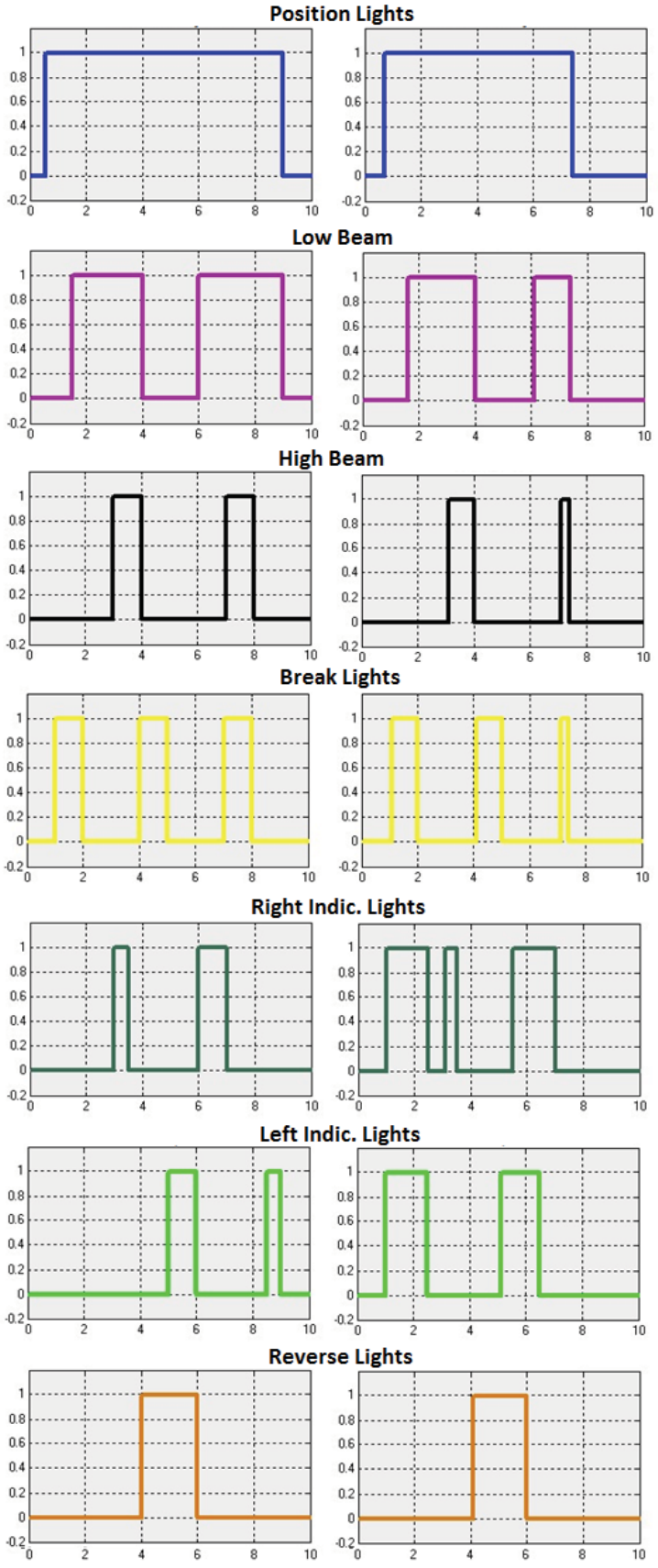

FIGURE 6. Comparison between inputs (on the left) and outputs (on the right) to verify the controller's logic

Validation

- Battery is in a satisfying level from T = 0s to T = 730s;
- Ignition position is at least in Acc from T = 60s to T = 800s.

Position Light:
REQ-01 and REQ-02: Position light is active during the interval from T = 50s to T = 900s. The output shows that this case is active from T = 60s to T = 730s.
Dipped Beam:
REQ-03 and REQ-04: Dipped beam is active from T = 300s to T = 400s and from T = 700s to T = 800s. The outputs shows that this case is active from T = 300s to T = 400s and from T = 700s to T = 730s.
Full Beam:
REQ-05 and REQ-06: Full beam is active from T = 150s to T = 400s and from T = 850s to T = 900s. The outputs shows that this case is active from T = 150s to T = 400s and from T = 600s to T = 730s.
Left Indicator Light:
REQ-07 and REQ-08: Left indicator is active from T = 500s to T = 600s and from T = 850s to T = 900s. In addition, the warning-hazard is active from T = 100s to T = 250s and from T = 550s to T = 650s. The outputs shows that this case is active from T = 100s to T = 250s and from T = 500s to T = 650s.
Right Indicator Light:
REQ-09 and REQ-10: Right indicator is active from T = 300s to T = 350s and from T = 600s to T = 700s. In addition, the warning-hazard is active from T = 100s to T = 250s and from T = 550s to T = 650s. The outputs shows that this case is active from T = 100s to T = 250s, from T = 300s to T = 350s and from T = 550s to T = 700s.
REQ-11: As observed when the hazard warning is active, both indicator lights are active.
Reverse Light:
REQ-12 and REQ-13: Full beam is active from T = 400s to T = 600s. The outputs shows that this case is active from T = 400s to T = 600s.
Brake Light:
REQ-14: Brake pedal is active from T = 100s to T = 200s, from T = 400s to T = 500s and from T = 700s to T = 800s. The outputs shows that this case is active from T = 100s to T = 200s, from T = 400s to T = 500s and from T = 700s to T = 730s.
Battery:
REQ-15: All cases are only operation from T = 0s to T = 730s.

In order to perform RCP simulations, the same test case was used, but since Simulink® does not allow the Signal Builders block to run repeatedly, the inputs were replicated in LabView®. In order to connect with the MicroAutoBox®, a Data Acquisition hardware from National Instruments was used. In order to provide a better illustration of the comparison between inputs and outputs, the cases were divided into two groups. The first group included the Position Lights case, the Low Beam case, the High Beam case and the Brake Light case. The second group included both Indicator Lights cases and the Reverse Light case. Figure 9 compares the first group of test case inputs with the same case MicroAutoBox® outputs, and Figure 10 compares the second group of inputs and outputs for RCP.

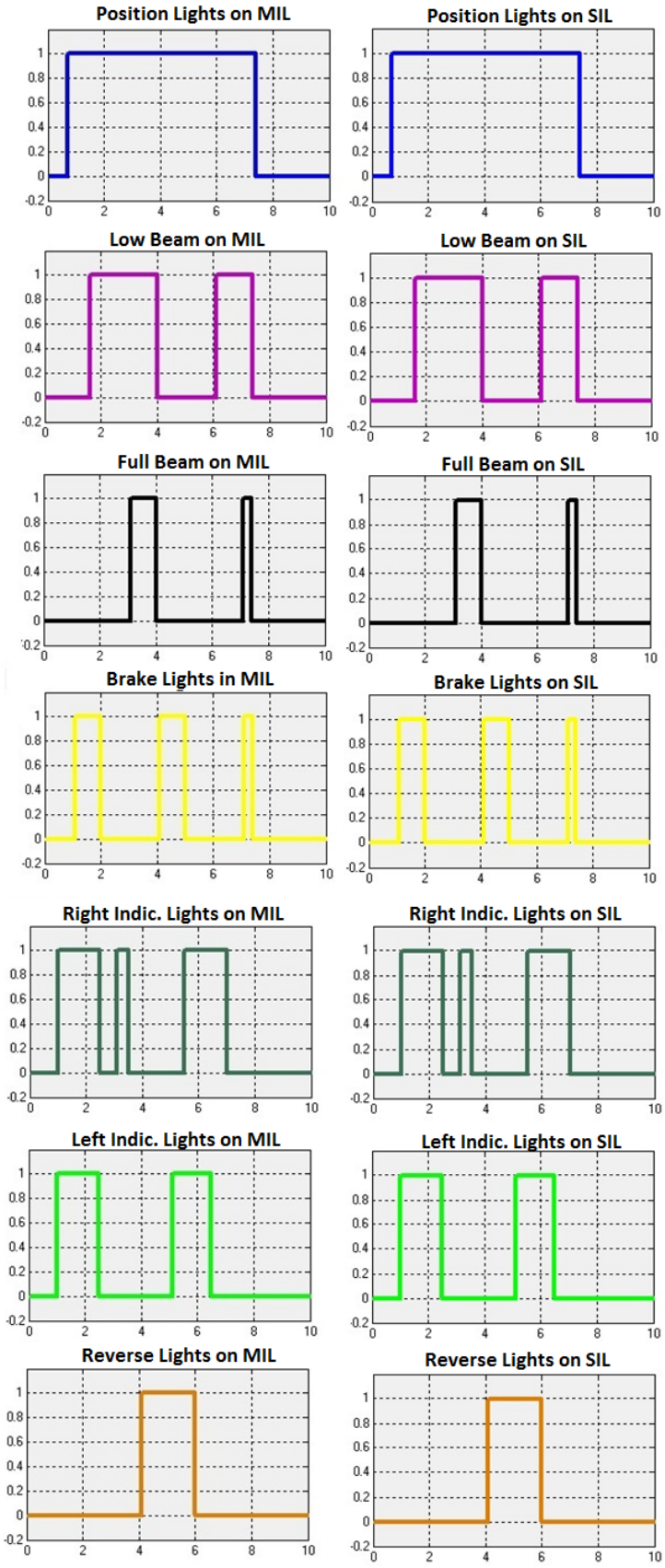

FIGURE 7. Comparison between Model-In-the-Loop (MIL) and Software-In-the-Loop (SIL) outputs

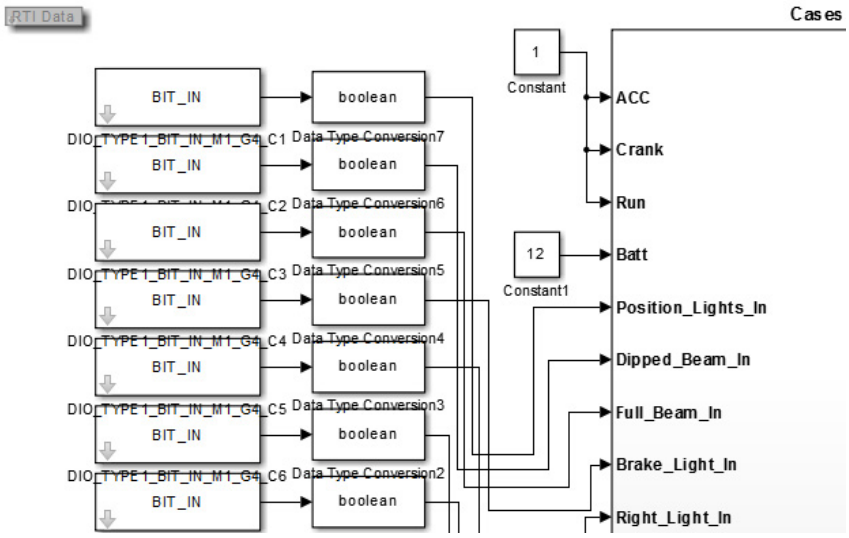

FIGURE 8. Modified plant and controller model on Simulink® to run in MicroAutoBox®

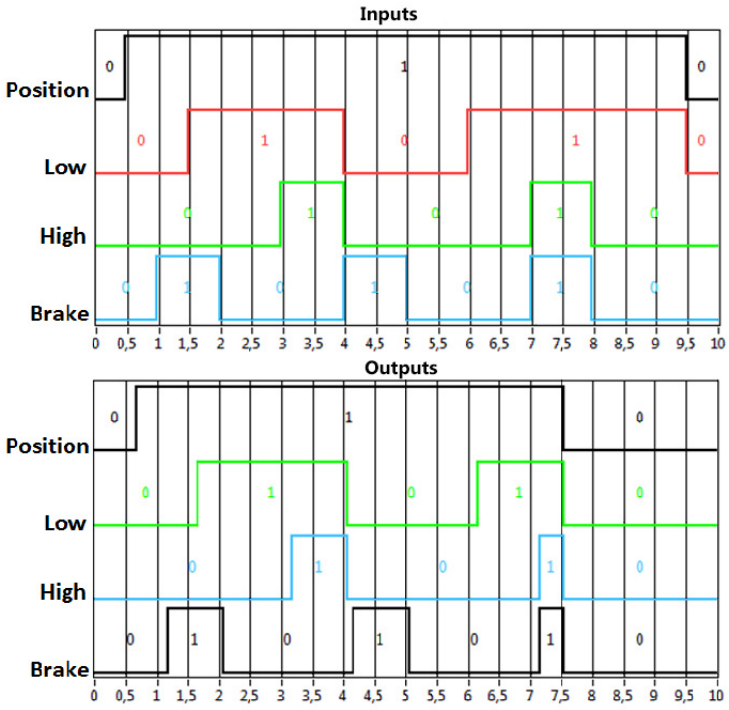

FIGURE 9. Comparison between the first group of inputs and outputs for RCP

This comparison allowed concluding that, equivalent to the phases of Model-In-the-Loop and Software-In-the-Loop, the system behaved correctly. The requirements were all met, including battery level and the Hazard Warning.

7. **Conclusion.** In this study, the control design and implementation for an External Lighting System are largely supported by Model Based Design via Matlab®/Simulink® and rapid control prototyping system using MicroAutoBox® from dSpace. The performance of the system was easily tested using those tools with computer simulations. Model Based Design offered a significant improvement compared to the traditional approach and RCP proved to be a powerful development tool.

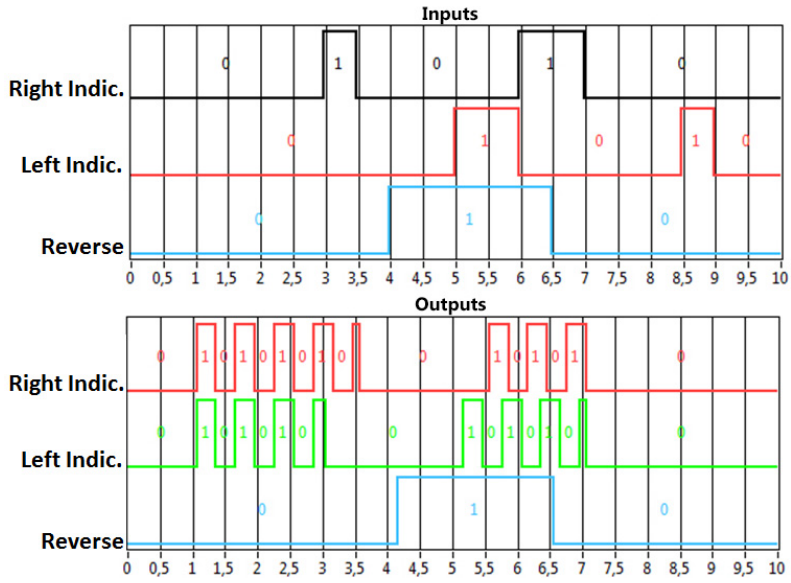

FIGURE 10. Comparison between the second group of inputs and outputs for RCP

Another advantage in this process is demonstrated in the Software-In-the-Loop phase. The automatic coding can also save development time and due to MBS characteristics, if requirements are not met, it is possible to perform changes in the Simulink® model and a new C code could be built rapidly.

REFERENCES

[1] H. Rajeshwari, G. Mishra and K. S. Gurumurthy, An insight into the hardware and software complexity of ecus in vehicles, *Advances in Computing and Information Technology*, pp.99-106, 2011.

[2] A. Sangiovanni and M. D. Natale, Embedded system design for automotive applications, *IEEE Computer*, vol.40, no.10, pp.42-51, 2007.

[3] M. Brou, S. Kirstan, H. Krcmar and B. Schatz, What is the benefit of a model-based design of embedded system in the car industry? *Emerging Technologies for the Evolution and Maintenance of Software Models*, pp.343-369, 2012.

[4] E. Eyisi, Z. Zhang, X. Koutsoukos, J. Porter, G. Karsai and J. Sztipanovits, Model-based control design and integration of cyber-physical systems: An adaptive cruise control case study, *Journal of Control Science and Engineering*, 2013.

[5] B. Guilherme, *Definition and Classification of Faults of an Automotive Electrical/Electronic System Based on the Standard ISO 26262*, 2012.

[6] X. Dong, M. Yu, C. Liao and W. Chen, Rapid control prototyping development of intelligent control system of vehicle semi-active suspension, *Proc. of the 7th World Congress on Intelligent Control and Automation*, Chongqing, China, 2008.

[7] I. Fey, J. Muller and M. Conrad, Model-based design for safety-related applications, *SAE World Congress 2008*, Detroit, USA, 2008.

[8] S. M. Prabhu and P. J. Mosterman, Model-based design of a power window system: Modeling, simulation and validation, *Proc. of IMAC-XXII: A Conference on Structural Dynamics*, Dearborn, MI, 2004.

[9] M. Conrad, P. Munier and F. Rauch, *Qualifying Software Tools According to ISO 26262*, http://www.mathworks.com/tagteam/61793_CMR10-16.pdf, 2014.

[10] M. Conrad, *Verification and Validation According to ISO 26262: A Workflow to Facilitate the Development of High-Integrity Software*, http://www.mathworks.com/tagteam/71300_1D-4.pdf, 2014.

[11] M. K. Lee, S.-H. Hong, D.-C. Kim and H. M. Kwon, Incorporating ISO 26262 development process in DFSS, *Proc. of the Asia Pacific Industrial Engineering & Management Systems Conference*, 2012.

[12] *Official Website of the Mathworks: www.mathworks.com.*