WHITE PAPER

Developing Battery Management Systems with Simulink and Model-Based Design

MathWorks®

Across industries, the growing dependence on battery pack energy storage has underscored the importance of battery management systems (BMSs) that can ensure maximum performance, safe operation, and optimal lifespan under diverse charge-discharge and environmental conditions. To design a BMS that meet these objectives, engineers develop feedback and supervisory control algorithms that:

- Monitor cell voltage and temperature
- Estimate state-of-charge (SOC) and state-of-health (SOH)
- Limit power input and output for thermal and overcharge protection
- Control the battery charging profile
- Balance the state-of-charge of individual cells
- Isolate the battery pack from source and load when necessary

This paper describes how engineers develop BMS algorithms and software by performing system-level simulations with Simulink®. Model-Based Design with Simulink enables you to gain insight into the dynamic behavior of the battery pack, explore software architectures, test operational cases, and begin hardware testing early, reducing design errors. With Model-Based Design, the BMS model serves as the basis for all design and development activities, including desktop simulation of the design's functional aspects, formal verification and validation to industry standards, and code generation for real-time simulation and hardware implementation (Figure 1).

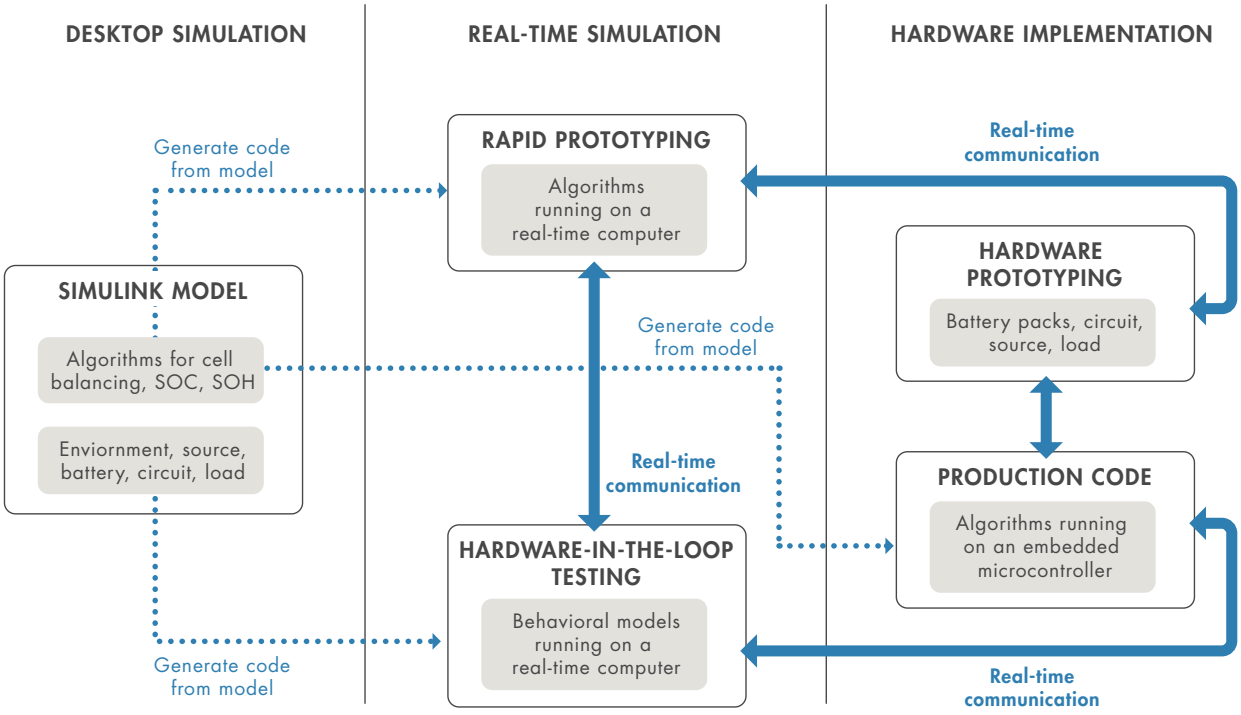

Figure 1. Battery management system development workflow with Simulink and Model-Based Design.

Desktop Simulation. Desktop simulations in Simulink enable you to verify functional aspects of the BMS design, such as charge-discharge behavior (using single-cell equivalent circuit formulation), electronic circuit design, and feedback and supervisory control algorithms. On the desktop, the battery system, environment, and algorithms are simulated using behavioral models. For example, you can explore active vs. passive cell balancing configurations and algorithms to evaluate the suitability of each balancing approach for a given application. You can use desktop simulation to explore new design ideas and test multiple system architectures before committing to a hardware prototype. You can also perform requirements testing in desktop simulations, for example by verifying that contactors are prevented from opening or closing when an isolation fault is detected.

Real-Time Simulation. Once validated via simulation, Simulink models can be used to generate C and HDL code for rapid prototyping (RP) or hardware-in-the-loop (HIL) testing to further validate the BMS algorithms in real time. With RP, instead of handwriting control software code for real-time testing, you generate code from your controller model and deploy it to a real-time computer that performs the functions of the production microcontroller. With automatic code generation, algorithm changes made in the model can be tested on real-time hardware in hours rather than days. Further, you can interact with real-time control hardware from within Simulink to change algorithm parameters and log test data.

As with rapid prototyping, HIL testing involves generating code from a Simulink model and deploying it to a real-time computer. In the case of HIL testing, code is generated from the battery system models rather than the control algorithm models, providing a virtual real-time environment that represents battery pack, active and passive circuit elements, loads, charger, and other system components. This virtual environment lets you validate the functionality of the BMS controller in real time before developing a hardware prototype and in an environment where hardware will not be damaged. Tests developed during desktop simulation can be carried over to HIL testing, to ensure that requirements are met as the BMS design progresses. Though HIL testing is employed primarily to test code running on a microcontroller or FPGA, you can instead use a rapid prototyping system, such as ***Simulink Real-Time***™ and ***Speedgoat target hardware***, connected to the HIL setup before production controller hardware is selected.

Hardware Implementation. In the hardware implementation stage, the Simulink control models that have been verified via desktop simulation, RP, and HIL are used to generate efficient, production-ready code for the BMS. If necessary, production code generation can be incorporated into workflows compliant with formal certification standards used in the automotive, aerospace, and other regulated industries.

> *"Modeling and simulation with MATLAB, Simulink, and Simscape™ is faster, safer, and less costly than building physical prototypes. We can identify algorithms or charging methods that will work for a particular design without running the whole system. We can test scenarios that would be difficult or hazardous to test on real batteries and optimize designs for specific applications and usage profiles."*
>
> **— Cecilia Wang, Romeo Power**

» ***Read story***

Desktop Simulation: Modeling BMS Software

The ability to perform the realistic simulations that are central to the development of BMS control software starts with an accurate model of the battery pack. Batteries are often designed using finite element analysis (FEA) models that account for the physical configuration of the batteries and capture their electro-thermochemical properties. Although these models are excellent for designing and optimizing a battery pack's chemistry and geometry, control engineers need models that are better suited for system-level design and software development.

> *"Many of our initial battery models were empirical, with an ideal voltage source and a fixed impedance. We now use much more sophisticated first-principles models, and as a result, have gained invaluable insights into the battery as an electrochemical device. We used Simulink to build advanced equivalent circuit models that can predict performance at different states of charge, discharge rates, temperatures, and levels of aging.*
>
> *"We used a similar approach to perform safety-critical simulations to predict cooling performance within the battery and ensure that the battery packs would not overheat. To capture all the multidomain physical, chemical, and heat-transfer effects would typically require a finite-element analysis tool and significant effort. With MathWorks tools we performed analyses and gained insights that lead to dramatic advances in battery technology."*
>
> **— Dr. Chris Gadda and Dr. Andrew Simpson, Tesla Motors**

» ***Read article***

Modeling and Characterizing the Battery Cell

When developing BMS algorithms in Simulink, you can use equivalent circuits to simulate the thermo-electric behavior of the battery cell. The equivalent circuit typically comprises a voltage source, a series resistance, and one or more resistor-capacitor pairs in parallel (Figure 2). The voltage source provides the open circuit voltage while the other components model the internal resistance and time-dependent behavior of the cell. These equivalent circuit elements are, in general, temperature and state of charge (SOC) dependent. Because these dependencies are unique to each battery's chemistry, they need to be determined using measurements performed on battery cells of the same type as those for which the controller is being designed. You can use optimizations in Simulink and MATLAB® to parameterize an equivalent circuit via ***model correlation with experimental data***.

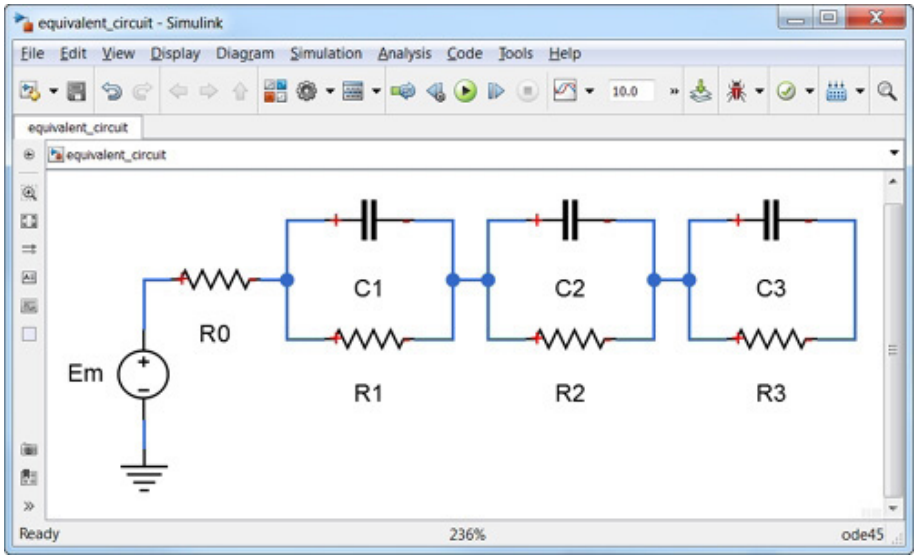

Figure 2. Equivalent circuit of a battery with three distinct time constants, internal resistance, and open circuit potential.

By incorporating ***Simscape Electrical***™ components, you can scale up from the unit cell level to the module and pack level and intuitively combine cells with surrounding circuitry. With pack-level simulation you can evaluate the effects of various pack configurations on energy storage capacity, power delivery rates, and thermal operational envelope. Pack-level simulations also let you explore the pack's interaction with other system components such as source, load, and protection circuits.

Learn More About Modeling and Characterizing the Battery Cell

- *Lithium Battery Cell - Two RC-Branch Equivalent Circuit* - Example
- *Lithium-Ion Temperature Dependent Battery Model* - Example
- *Battery Models* - File Exchange
- *Parameterization of a Rechargeable Battery Model* - Example
- *Automating Battery Model Parameter Estimation* (9:55) - Video
- *Battery Model Parameter Estimation Using a Layered Technique: An Example Using a Lithium Iron Phosphate Cell* - Technical Paper

Modeling the Power Electronics and Passive Components

In addition to the battery pack model, realistic BMS simulations require accurate models of the circuit components connecting the battery system to the power source and load. Simscape Electrical, an add-on product for Simulink, provides complete libraries of the active and passive electrical components needed to assemble a complete battery system circuit, such as the analog front end for cell balancing. The charging source can consist of a DC supply, such as a photovoltaic (PV) system, or an AC source, for which the current is rectified.

System-level simulation with Simulink lets you construct a sophisticated charging source around the battery and validate the BMS under various operating ranges and fault conditions. The battery pack load can be similarly modeled and simulated. For example, the battery pack may be connected through an inverter to a permanent magnet synchronous motor (PMSM) in an electric vehicle (EV). With simulation, you can vary the operation of the EV through drive cycles and evaluate the effectiveness of the BMS in coping with changing operating conditions.

Learn More About Modeling the Power Electronics and Passive Components

- *High-Voltage Battery Feeding the IPMSM Through a Controlled Three-Phase Inverter* - Example

Developing Supervisory Control Algorithms

Engineers who develop BMS supervisory control algorithms use state machines to model supervisory logic for fault detection and management, charge and discharge power limitation, temperature control, and cell balancing (Figure 3). ***Stateflow®*** is an environment for modeling and simulating combinatorial and sequential decision logic based on state machines and flow charts. When developing supervisory control algorithms for a BMS, you can use Stateflow to model how the battery system reacts to events, time-based conditions, and external input signals. For example, in the case of constant current constant voltage (CCCV) charging, you can develop and test the state logic that controls when the cell transitions from current charging mode to voltage charging mode.

> *"With Model-Based Design we have an integrated process for development, from idea through production code generation. MathWorks tools enabled us to develop key battery management technology using our own expertise, in an environment that facilitated early and continuous verification of our design."*
>
> **— Dr. Xiaokang Liu, Dongfeng Electric Vehicle**

» ***Read story***

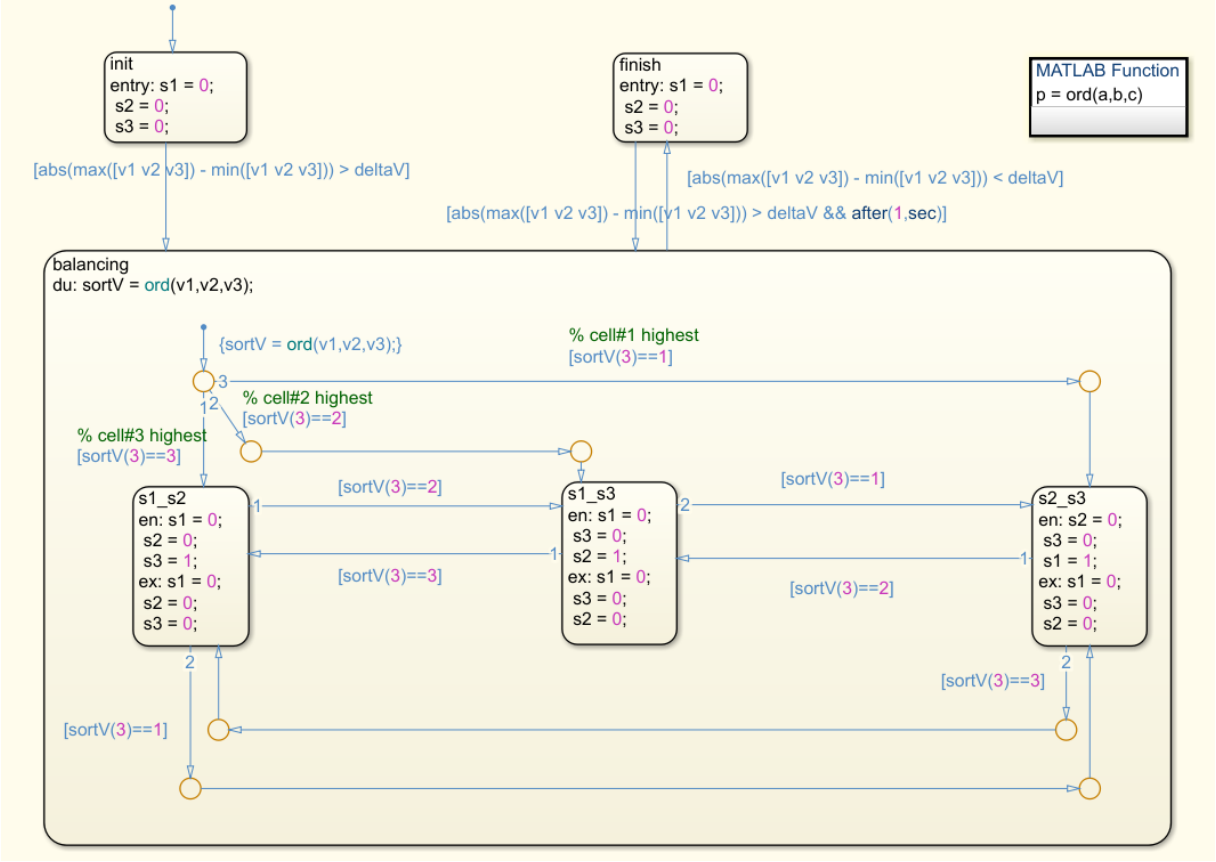

Figure 3. Cell balancing logic implemented in a Stateflow diagram.

Learn More About Developing Supervisory Control Algorithms

- *Battery Management System Development in Simulink* (7:18) - Video

Estimating State of Charge

Accurate battery models are vital in the development of algorithms for SOC estimation. Traditional approaches to SOC estimation, such as open-circuit voltage (OCV) measurement and current integration (coulomb counting), are reasonably accurate in some cases. However, estimating the SOC for modern battery chemistries that have flat OCV-SOC discharge signatures requires a different approach. Extended Kalman filtering (EKF) is one such approach that has been shown to provide accurate results for a reasonable computational effort. Simulink contains an EKF block that enables you to develop an observer for estimating SOC. Such observers typically include a model of the nonlinear system of interest (the battery), which uses the current and voltage measured from the cell as inputs, as well as a recursive algorithm that calculates the internal states of the system (SOC among them) based on a two-step prediction/update process (Figure 4).

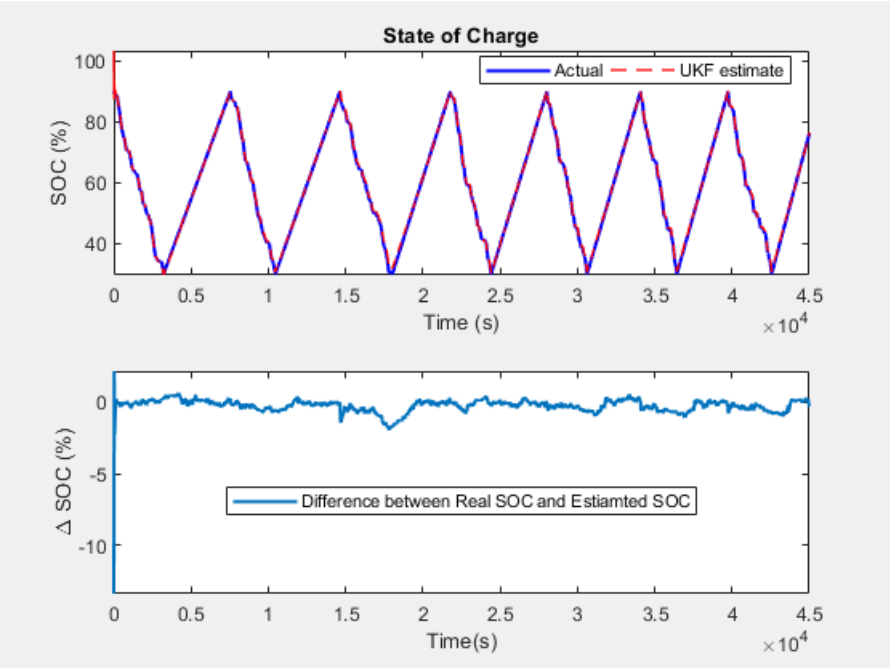

Figure 4. Estimating battery state of charge using an unscented Kalman filter in Simulink.

Learn More About Estimating State of Charge

- *State of Charge (SoC) Estimation Based on an Extended Kalman Filter Model* - Article
- *Battery Management System Reference Design* - Intel Documentation
- *Nonlinear State Estimation of a Degrading Battery System* - Example
- *Extended Kalman Filter* - Documentation

Estimating State of Health

All batteries, including those that meet performance specifications at time of manufacture, degrade over time due to calendar life and cycling, suffering a gradual loss in reserve capacity and an increase in internal resistance. While the latter is relatively straightforward to estimate using short time measurements, the former requires a full charge or discharge excursion for an accurate calculation, which is not always practical. This challenge has led to growing interest in state of health (SOH) estimation as well as the development of EKF formulations augmented to include battery parameters in addition to states. An accurate estimation of the instantaneous internal resistance is very helpful for the BMS to establish power limitations.

SOH estimation is more subjective than SOC estimation; there is no universal agreement on how SOH is to be defined. As a result, each organization may have its own specific method for quantifying an SOH estimate, making it impossible to use general-purpose, off-the-shelf solutions. With Simulink, you can develop and simulate custom SOH estimation algorithms that are in line with your organization's specific interpretation of battery health.

Learn More About Estimating State of Health

- *Model-Based Parameter Identification of Healthy and Aged Li-ion Batteries for Electric Vehicle Applications* - Technical Paper

Testing with Desktop Simulation

When you incorporate desktop simulation into your testing procedures, you can author and execute test cases to exercise the BMS along all possible branches of logic and closed-loop control—a level of coverage rarely available when testing with hardware. With this approach, the simulation model serves as an executable specification driving the design and testing of the BMS (Figure 5). Simulink supports testing via desktop simulation with a range of features that enable you to:

- Incorporate requirements, such as limits, tolerances, logical checks, and temporal conditions, into the model with traceability to the original specifications
- Construct complex sequences of simulation-based tests to perform functional, baseline, equivalence, and back-to-back testing
- Track industry-standard metrics such as decision, condition, and modified condition/decision coverage (MC/DC), as well as relational boundary coverage
- Generate test inputs to achieve complete model coverage and custom objectives
- Use formal methods to identify hidden design errors that result in integer overflow, dead logic, and division by zero

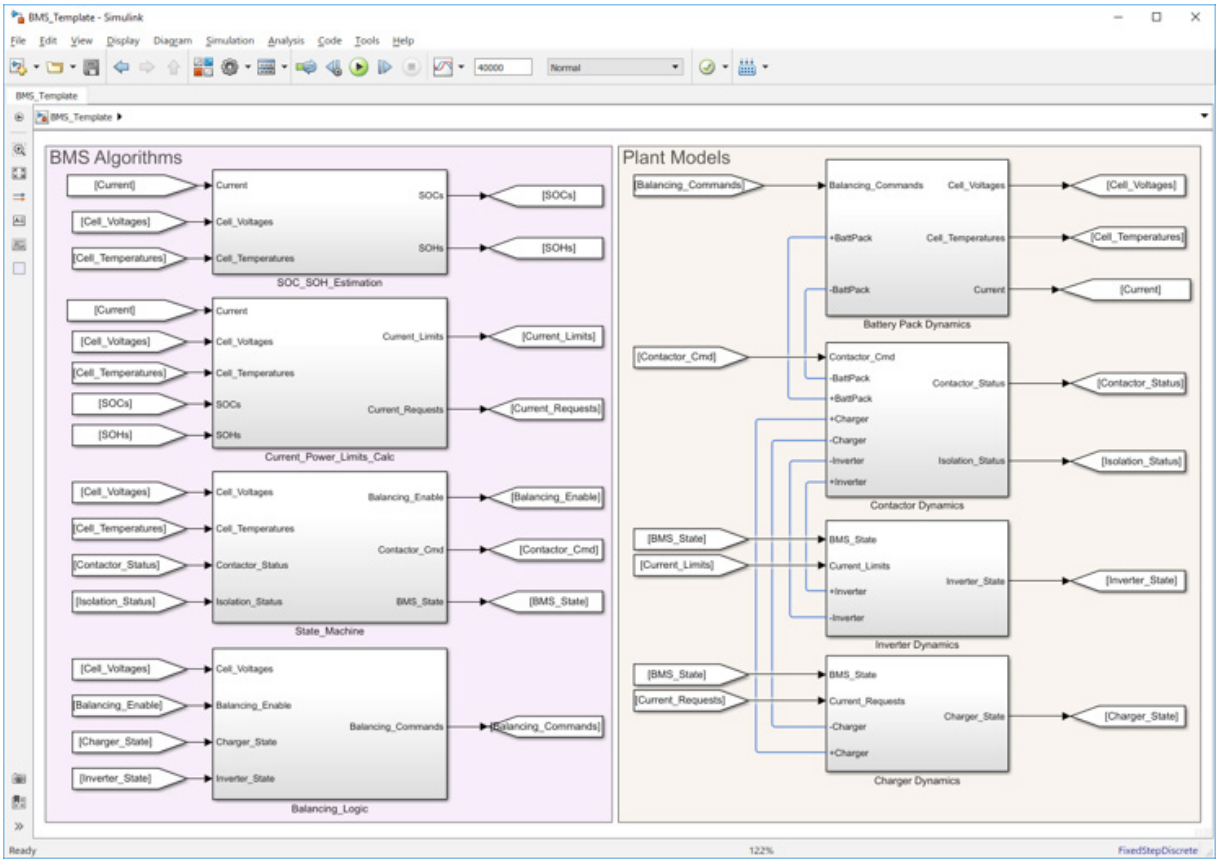

Figure 5. BMS algorithms and plant dynamics, including battery pack, contactor, inverter, and charger, modeled in Simulink.

When the battery system under development must meet safety requirements, you can integrate tests based on formal methods into your software development process in accordance with standards such as IEC 61508, IEC 61851, and ISO 26262.

> *"While developing the controller model in Simulink, we frequently checked compliance with style guidelines and modeling standards using the Model Advisor. We also checked for dead logic, divide-by-zero errors, and other design errors in our models using Simulink Design Verifier™."*
>
> **— Duck Young Kim, Won Tae Joe, and Hojin Lee, LG Chem**

» ***Read article***

Learn More About Testing with Desktop Simulation

- *Developing AUTOSAR- and ISO 26262-Compliant Software for a Hybrid Vehicle Battery Management System with Model-Based Design* - Article
- *Using Model-Based Design to Build the Tesla Roadster* - Article

Real-Time Simulation: Validating BMS Software

Real-time simulation—encompassing both rapid prototyping and HIL testing—provides power electronics control engineers with additional insights into how a BMS design will perform on hardware. In both RP and HIL, the objective is to emulate in hardware one aspect of the overall design: the BMS controller in RP and the balance of the battery system in HIL. Real-time simulation offers several significant advantages in BMS design, letting you:

- Conduct RP to start validating algorithms before the final controller hardware is selected
- Exploit the flexibility of a real-time test system for rapid design iteration and testing
- Conduct HIL testing before the battery system prototype hardware is available
- Use a combination of RP and HIL testing to exercise BMS algorithms for test cases that may be difficult, expensive, or destructive if you were to use the actual hardware (Figure 6)

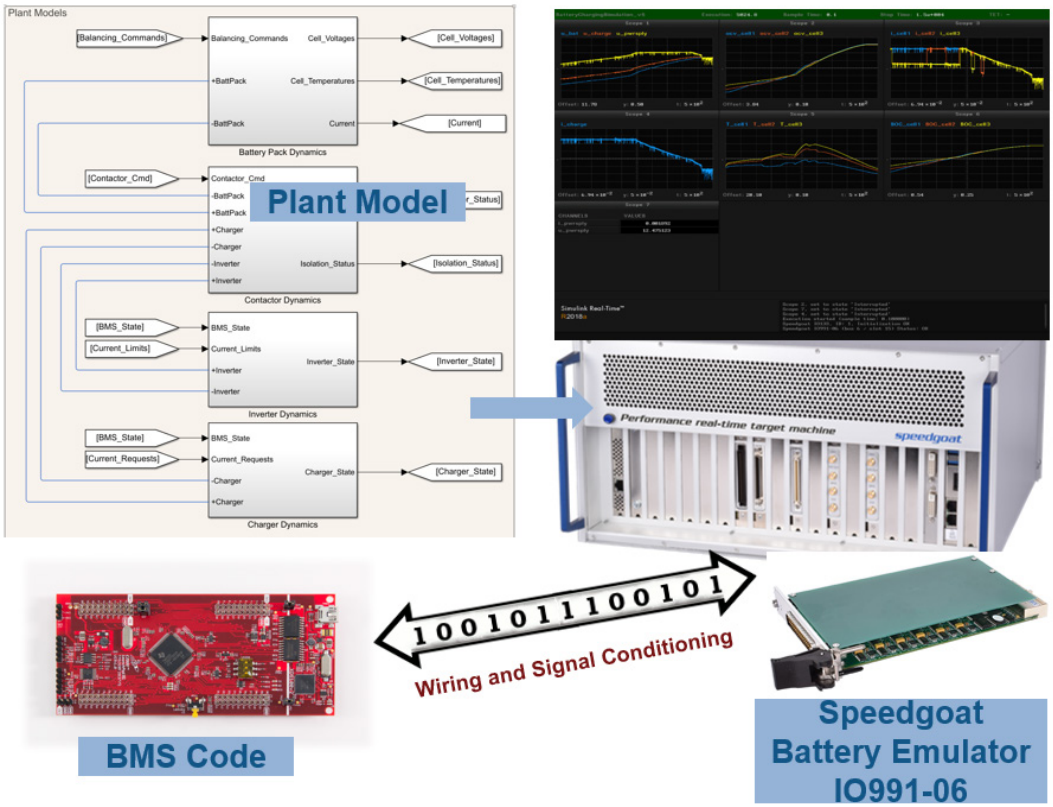

Figure 6. Hardware-in-the-loop (HIL) testing of battery management system software. The BMS code is generated from BMS algorithms modeled in Simulink and deployed to Texas Instruments C2000 microcontroller. The plant model (battery pack, contactor, inverter, charger) is modeled in Simulink. Code is generated and deployed to run on Speedgoat real-time machine with battery emulator.

By reusing desktop simulation models in Simulink to generate code for real-time simulation, you can shorten overall development time. You can generate C/C++ and HDL code that executes on computers optimized for real-time performance. Code generated from Simulink models for real-time simulations includes interfaces that enable you to adjust control parameters while the real-time simulation is running.

Performing Rapid Prototyping

During hardware testing, making changes to controller code can cause delays and additional risks. Modifying the code by hand, recompiling it, and deploying it to the microcontroller or FPGA takes time—potentially a long time if you are a control algorithm developer who relies on a software or hardware engineer to make the changes. Depending on the extent of the changes required, you also risk introducing new problems into the implemented code.

Instead of handwriting code updates to controller software, you can use Simulink to generate code that executes in real time on a dedicated computer and uses high-speed I/O to communicate with test hardware. In addition to eliminating manual coding and its associated delays, another advantage of this RP approach is that you can validate changes to the BMS software by running the simulation model on the desktop first to verify that no other problems were introduced.

Testing with Hardware-in-the-Loop

Because hardware prototypes for a battery system can take considerable effort to build and modify, and because they are often costly to repair, it is not always feasible to test such prototypes against the electrical system in which the battery pack will operate. Given these limitations, even small design changes can threaten development schedules, and BMS designs tend to evolve slowly because teams consider radical departures from the previous design as too risky.

With Simulink, you can generate C/C++ and HDL code from the model of the hardware in your battery system and the greater system of which it is part, including the supply and the load. Once you deploy this code to a real-time computer, you are able to run real-time simulations of the hardware against your controller code before testing the controller in a battery system prototype. As a result, you can find and correct control design errors before they potentially damage expensive and difficult-to-replace prototype hardware. You can also uncover hardware design errors, such as incorrect component sizing.

Many HIL real-time systems, including Speedgoat target hardware, incorporate battery emulators, letting you emulate portable battery power supplies, emulate battery stacks for electric vehicles or sink current to simulate batteries under charge.

> *"I feel that Speedgoat has certainly developed a plug-and-play real-time platform for Simulink. For us, that translates into more time testing our control systems and less time developing a HIL bench."*
>
> **— Joaquin Reyes, Controls Engineer, Proterra**

» ***Read story***

Learn More About Testing Battery Management Systems with Hardware-in-the-Loop

- *Speedgoat Real-Time Solutions for Battery Management Systems* - Overview

Production-Ready Code Generation

Simulink generates readable, compact, and efficient C/C++ and HDL code from controller models that is ready for implementation on production microcontrollers, FPGAs, and ASICs. Unlike code generated for RP, code generated for production use does not include the extra interfaces needed to support real-time monitoring, parameter tuning, and data logging. Optimization settings enable you to precisely control the generated functions, files, and data to improve code efficiency and facilitate integration with legacy code, data types, and calibration parameters.

Performing Processor-in-the-Loop Simulations

In processor-in-the-loop (PIL) simulation, the C/C++ or HDL code runs on the microcontroller or FPGA while the device is stepping in execution with a Simulink model of the BMS hardware, limiting the risk of damaging a hardware prototype during initial evaluations of the BMS code. Although PIL simulations are not executed in real time, they are bit-true, enabling you to verify your code under a range of conditions and build confidence that it will execute properly once deployed on the real system.

Generating Production Code

Desktop simulation, RP, HIL, and PIL simulations all enable you to verify and validate the control algorithms for the BMS. With Simulink, you can use those same algorithms as the basis for generating production-ready code—either optimized and stable C/C++ code for implementation on microcontrollers or synthesizable HDL code for FPGA programming or ASIC implementation. Automatic code generation eliminates manual algorithm translation errors and produces C/C++ and HDL code with numerical equivalence to the algorithms you validated in Simulink. By simulating your control algorithms over all possible operating and fault conditions, you increase confidence that the generated code will handle those same conditions in the real system, even if you are unable to test for all of them. If hardware tests later indicate that algorithm changes are needed, you can simply modify the algorithms in your model, rerun simulation test cases to verify the correctness of the changes, and generate new, updated code. All generated C/C++ and HDL code is fully portable, optimizable with a range of options, and bidirectionally traceable to the Simulink model (Figure 7).

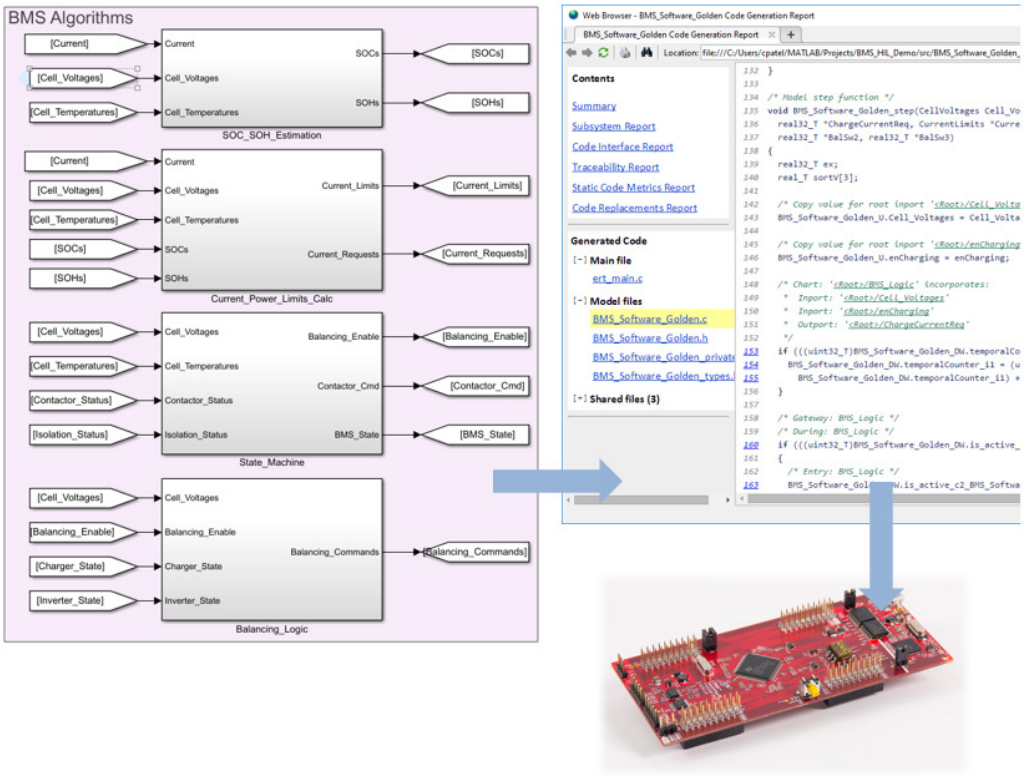

Figure 7. Automatically generating BMS production code from BMS algorithms modeled in Simulink. Code is deployed to Texas Instruments C2000 microcontroller.

"We check every requirement needed for certification via simulation in Simulink before verifying it on the real circuit—and because we used Embedded Coder to generate code directly from our model, there's no gap between our simulations and the real embedded software."

— Dr. Yue Ma, Murata Manufacturing Co., Ltd.

» ***Read story***

Learn More About Generating Production Code

- *Murata Manufacturing Reduces Development Time for Energy Management System Control Software by More Than 50% with Model-Based Design* - User Story
- *Developing AUTOSAR- and ISO 26262-Compliant Software for a Hybrid Vehicle Battery Management System with Model-Based Design* - Article

Next Steps

Take the next step to speed up your battery management system project.

Explore: *Battery Management Systems in Simulink*

Download: *Trial Software for Motor and Power Control*

Get Started: *Consulting Services*