EVS27
Barcelona, Spain, November 17-20, 2013

Design of a Plugin Split-Parallel Chevrolet Malibu: Control Strategy Development

Marc Herniter [1], Zachariah Chambers, Jon Nibert

[1]*Rose-Hulman Institute of Technology, 5500 Wabash Avenue, CM123, Terre Haute, IN 47803*

herniter@rose-hulman.edu

Short Abstract

As a selected participant in the EcoCAR2: Plugging In to the Future competition, headline sponsored by General Motors and the US Department of Energy, the Rose-Hulman team has completed the second year of a three year design cycle. This paper focuses on the supervisory control system design completed during year one of the competition including plant model development, control system hardware architecture, control system software architecture, vehicle operational modes, automated testing, and fault detection and mitigation.

1 Introduction

Rose-Hulman Institute of Technology (RHIT) is an undergraduate focused college of engineering, science, and mathematics located in Terre Haute, Indiana USA.[1] RHIT has been participating in the General Motor (GM) and US Department of Energy (DoE) headline sponsored Advanced Vehicle Technology Competitions (AVTCs) since 2004 when selected to join the ChallengeX competition. These AVTCs are rigorous three year design cycles which challenge 15 North American universities and colleges to design, deploy, and validate a hybrid vehicle utilizing alternative fuels to decrease petroleum consumption and emissions production on a Well-to-Wheels basis without compromising consumer acceptability in performance, utility, and safety.[2] The first year focuses on vehicle system modelling utilizing Matlab and Simulink to predict performance specifications in parallel with investigating component fitment via Siemens NX 7.5. Year two culminates with the testing of a 60% buyoff mule vehicle which is then optimized and refined to 99% buyoff quality during year three. For the current AVTC EcoCAR2: Plugging In to the Future the vehicle platform is a 2013 Chevrolet Malibu.

A key team decision was to create a performance oriented vehicle as a market differentiator: the Malibu eco^{sport}. Modeling and simulation work performed during year one, within the constraints of available GM or third party resources, indicated that a plugin split-parallel architecture would best meet the team and competition goals. The components selected are presented in Table 1 and high-level architecture layout is presented in Figure 1.

Component	Source
Engine	GM LE9 2.4l four cylinder (using E85)
Transmission	GM MHH 6-speed automatic with eAssist
Belt Electric Machine	Modified GM BAS, Rinehart Inverter
Rear Traction Motor	Remy HVH250p
Energy Storage System (ESS)	A123 blah blah

Table 1: Selected Powertrain Components

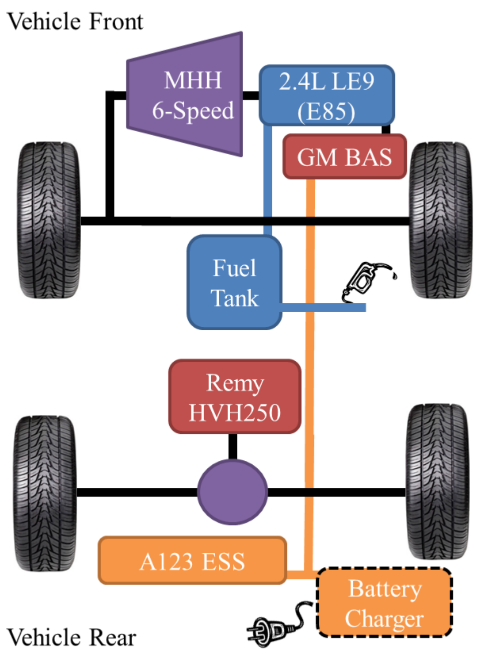


Figure 1: the RHIT Split-Parallel Chevrolet Malibu architecture.

The control strategy has two primary operating modes: charge depleting and charge sustaining.[3] In charge depleting mode, the vehicle operates electrically, powered by the rear traction motor. When the energy storage system (ESS) drops to a specified minimum threshold, the vehicle switches to an all-wheel drive charge-sustaining hybrid-electric vehicle. The BAS will be employed during year three as a refinement to enable engine auto-stop functionality. To achieve the performance goal of the Malibu eco^{sport}, a high throttle request from the driver overrides the current mode and activates both the engine and the motor for power levels approaching 330 hp while still achieving a combined cycle fuel economy of 41 MPGGE.

2 Vehicle Operational Modes

Several modes of operation have been built in to the controller. These modes include testing modes as well as operational modes. Testing modes are built into the controller to allow the control system to be tested incrementally, while operational modes allow the torque sources to be combined in different methods to achieve different performance criteria. All of the modes and features listed below have been implemented into the control system and tested with MIL, SIL, and HIL.

The following test modes have been created:

- Manual Test – In this mode, each component is controlled independently through MotoHawk overrides.[4] Status signals sent by the component are received by the HVSC and control signals are sent from the HVSC to the component. Torque requests are manually tested. Component speeds and directions can be verified.
- Engine Only – In this mode, the entire vehicle can be operated with the engine only. The ESS, and traction and BAS motors are disabled. This mode allows testing of the engine, transmission, startup and shutdown procedures, and vehicle shifting.
- Traction Motor Only – The vehicle operates in EV mode with the Traction motor powering the rear wheels only. The mode allows testing of the Traction Motor, startup and shutdown procedures, and vehicle shifting.
- Engine + BAS – In this minimal hybrid mode, the BAS assist is used together with the engine, requiring the engine and HV system to be active at the same time. The Traction Motor is disabled. This mode allows testing of the engine, transmission, ESS, BAS motor, startup and shutdown procedures, and vehicle shifting. For the moment, the BAS motor only provides low-speed and stationary charging. Future improvements may use this motor to improve vehicle efficiency.
- Engine + Traction Motor – In this nearly full-featured hybrid mode, the traction motor and the engine are used to implement the hybrid strategy without the aid of the BAS Motor is disabled. Almost all of the hybrid functions of the vehicle can be tested in this mode except for low-speed and stationary charging.)
- Engine + Traction Motor +BAS – Full-Featured Hybrid operation. Note that if a fault occurs in this mode, the

offending component is usually shutdown, resulting in the vehicle operating in one of the reduced feature modes listed above.

The following hybrid modes of operation are available.

- Charge depleting (CD) mode with engine assist. - In CD mode most of the energy to propel the vehicle comes from the battery. The Traction Motor is the prime mover. For pedal requests below 50%, the engine provides no torque and the vehicle achieves the best fuel economy. As the pedal varies from 0 to 50%, the traction motor provides bet 0 and 100% of its available torque. (In this mode, the engine could be off or the engine could be idling and provide no torque. – See below for more details). For pedal requests above 50%, since the traction motor is at its maximum capacity, the engine fulfills the torque requests for the amount of pedal above 50%. (This will require the engine to turn on if auto on/off is enabled.) This mode allows the driver to have an electric-only CD mode vehicle, with power in reserve when needed.
- Charge Sustaining Mode – In this mode, the battery SOC is maintained within a narrow band of operation. For the moment, the mode of operation is very simple. When the battery has sufficient charge, it behaves like CD mode where the Traction Motor is the prime mover and the engine is available for high torque requests. When the SOC reaches a lower limit, the engine is the prime mover and the traction and BAS motors charge the battery. The engine is powerful enough to fulfill most driver torque requests. This is a simple strategy chosen for year 1. In the future, RHIT will incorporate strategies such as charge stealing where the battery is charged during high engine power requests where the engine operates most efficiently.
- High performance Mode – For high pedal requests, all available torque sources provide torque to accelerate the vehicle.

The Following efficiency features are implemented in the controller. Note that each feature can be enabled or disabled for testing and to bring up the control system incrementally.

- Pedal lift regen. When the driver releases the accelerator pedal, (and does not depress the brake pedal), the traction motor will apply a negative torque to slow the vehicle. The amount of torque starts at zero and ramps up over a few seconds. The amount of torque is speed dependent. This feature is effectively automatic regen braking to simulate the feel of a conventional vehicle when the accelerator pedal is released. The ramp time and magnitude are tunable, and the feature can be disabled.
- Auto-Neutral capability–The transmission is allowed to automatically shift into neutral if the following conditions occur: the feature is enabled, the engine is not required to charge the battery, and the driver throttle request is less than 50% for the requisite amount of time. If any of these conditions become false, the transmission automatically shifts out of neutral and back into drive. This feature significantly improved fuel economy.
- Engine on/off capability – The transmission is allowed to automatically shift into neutral if the following conditions occur: The feature

is enabled, the transmission is in Neutral, the engine is not required to charge the battery, and the driver throttle request is less than 50% for the requisite amount of time. If any of these conditions become false, the engine automatically turns on. This feature also improves fuel economy. [5]

All HEV control strategies will eventually become complex. An HEV control strategy will have several components: torque blending, vehicle start-up, vehicle shut down, charging, and fault detection and mitigation. Any of these systems, in itself, can be quite complex. All parts together can be daunting. Because of this complexity, RHIT uses the philosophy of start simple, add one function at a time, test, verify, and understand the consequences of the operation of that feature. When the new feature is understood within the context of the larger model, add another single feature and then repeat the process. With all of the different functions of the model and the control system, the plant model, controller, and all components eventually become very complex. The benefit to this approach is that students understand every block in the model. Thus, RHIT develops all of its own models for the EcoCAR2 competition.

As the model has grown more complex, it needs to be tested more thoroughly. Typically, when you add a new feature to the plant or controller, something new and strange happens. When this happens, typically a drive cycle and initial conditions which cause the condition to occur are identified. Since we need to test all of these conditions before be approve the system, scripts have been created to exercise the model through various conditions to see if the behaviours occur with the modified system. Scripts have been created for the VTS, general drive cycle testing, and fault testing. If a new fault is added to the system, a test is added to the fault script. If a new bad behaviour is observed, a new test is added to the drive cycle script. With these scripts, the new feature can first be manually tested to identify and newly created bad behaviour, and then auto-tested using the scripts to test for all of the previously known bad behaviours.

For now, the trade-offs are between complexity and performance. An example is the trade-off between engine always on transmission always in drive mode, engine always on and Auto-Neutral enabled, and Engine On/Off enabled and Auto-Neutral enabled. In many different respects, keeping the engine always on and keeping the transmission in drive all the time greatly simplifies many different parts of the vehicle realization (logic, A/C, 12 V). However, implementing these modes would greatly increase fuel economy. Thus it is not a trade-off in component costs versus fuel economy as much as it is a trade-off between fuel economy and how easy will it be to implement, understand, and make it the fuel saving features operational. Thus, many of the features implemented by RHIT can be turned off until we gain the understanding and maturity needed to implement and test the features.

3 Control Strategy Safety

The control system is verified by running hundreds of simulations. This involves running simulations manually as well as running scripts that exercise the model through a fixed set of conditions and drive cycles. Drive cycles must be run manually, because design engineers do not always know how to quantify a problem before they observe its effects. Once a problem is observed, the conditions that gave rise to the problem and a function that can detect the problem can be added to one of the scripts, so that it can be tested in the future. Thus far, three scripts have been created. The VTS script, the Fault Test script, and the Drive Cycle script.

- The VTS script runs the vehicle through the 4 ANL cycles in CD mode, runs the vehicle through the 4 ANL drive cycles in CS mode, and then runs two drive cycles that calculate acceleration times. The script then calculates several parameters required for the VTS. The drive cycle RMS error is calculated to verify that the vehicle could follow the drive cycle. Faults are also reported if they occur (they should not, but fault detection is active). Finally, all data and plots from all runs are saved for future inspection.) An example output from one of the VTS runs is shown below:

```
Directory Name: 08-09-2012 13_34_05
Model Name: MIL_10a

Run Comments
------------------------------
```

```
Fuel Tank Range Test

Run Options - Controller Setup
------------------------------
Controller_Sample_Time = 0.01.
Engine on/off operation is enabled.
Min Engine on time when Auto Engine On/Off is engaged = 10 (s).
Transmission Auto-Neutral operation is enabled.
Auto-N Min time Trans is required to be in D before Auto_N is engaged =
10 (s).
Drive cycle multiplier = 3.
Faults are disabled.


Results Summary
---------------




Charge Depleting Mode
__________________________
Electric Energy Per Mile = 307.178 (Wh/mi)
E85 Fuel Energy Per Mile = 83.3806 (Wh/mi)
Utility Factor = 0.649759.
Charge Depleting Range  = 44.218.
Petroleum Energy Use  = 36.7923. (Wh PE/mi)
Green House Gas Emissions (GHG)  = 220.814. (g/mi)
E85 Fuel Used in Charge Depleting Mode = 0.155435. (gal)
Average Battery Heating Power  = 403.226. (W)
Peak Battery Heating Power  = 2.61209. (kW)
Peak Battery Output Power  = 87.0696. (kW)
Peak Battery Current  = 261.055. (A)
Peak Battery Current, RMS 5 Second Window Average  = 218.868. (A)
Peak Battery Current, RMS 10 Second Window Average  = 190.237. (A)
Average Battery Discharge Current  = 33.4618. (A)
Time to discharge battery 100%  = 4217.35. (s)
Maximum Battery Voltage  = 355.159. (V)
Minimum Battery Voltage  = 331.16. (V)
Average Vehicle Speed  = 40.6939. (mph)
Drive Cycle Vehicle Speed rmsError  = 0.292887. (mph)

Charge Sustaining Mode
__________________________
Electric Energy Per Mile = -9.30179 (Wh/mi)
E85 Fuel Energy Per Mile = 1138.22 (Wh/mi)
Petroleum Energy Use  = 359.36. (Wh PE/mi)
Green House Gas Emissions (GHG)  = 291.047. (g/mi)
E85 Fuel Remaining after CD Mode = 8.84456. (gal)
CS Vehicle Range With Remaining Fuel = 184.317. (mi)
Average Battery Heating Power  = 157.76. (W)
Peak Battery Heating Power  = 2.35088. (kW)
Peak Battery Output Power  = 78.3628. (kW)
Peak Battery Current  = 249.659. (A)
Peak Battery Current, RMS 5 Second Window Average  = 225.638. (A)
Peak Battery Current, RMS 10 Second Window Average  = 189.116. (A)
Maximum Battery Voltage  = 343.351. (V)
Minimum Battery Voltage  = 312.641. (V)
Average Vehicle Speed  = 40.3665. (mph)
```

```
Drive Cycle Vehicle Speed rmsError  = 0.717677. (mph)


UF Corrected Summary
__________________________
Overall Energy Per Mile = 649.161 (Wh/mi)
E10 Equivalent mpg  = 51.8669. (MPGGE)
Petroleum Energy Use  = 149.769. (Wh PE/mi)
Green House Gas Emissions (GHG)  = 245.412. (g/mi)
Vehicle Range = 228.535. (mi)




VTS Summary
__________________________
Vehicle Simulation Mass [kg] = 2000.
0-60 mph Accel [s] = 6.17062.
50-70 mph Accel [s] = 4.38044.
Vehicle Range [km (miles)] = 367.793(228.535).
Charge Depleting Range [km (miles)] = 71.1621(44.218).
Charge Depleting Fuel Consumption [Wh/km] = 51.8102.
Charge Sustaining Fuel Consumption [Wh/km] = 707.254.
UF-Weighted Fuel Energy consumption [Wh/km] = 281.374.
UF-Weighted AC Electric Energy Consumption [Wh/km] = 121.996.
UF-Weighted Total Energy Consumption [Wh/km] = 403.369.
UF-Weighted WtW Petroleum Energy Usage [Wh/km] = 93.0619.
UF-Weighted WtW GHG Emissions [g/km] = 152.492.
Script Run Time = 3831.22 Seconds.
```

- The Fault Test script test every fault recognized by HVSC. For each fault, a drive cycle is run, a fault is inserted, and then the system response is observed. The script records if the controller detected the fault and responded. Presently, 30 faults are implemented, requiring the script to run 30 simulations. As new faults are added to the controller, the script will be updated. This script saves plots from all of the runs, so that the system response to a fault can be inspected after the script completes. Figure 2 shows an example plot of an accelerator pedal fault. When the fault is detected, the vehicle is shut down:

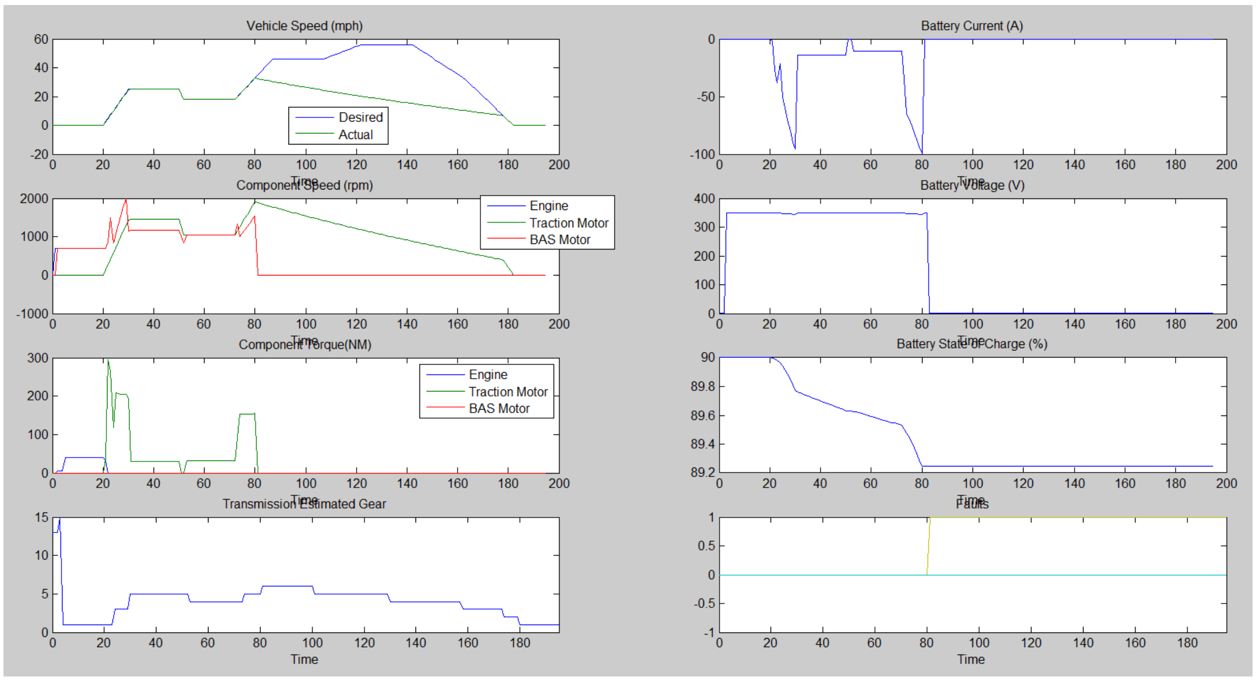


Figure 2: Vehicle response to an accelerator pedal fault.

The HVSC also records faults and has a status output for each fault. Figure 3 shows a fault status plot showing that the system detected an accelerator pedal fault.

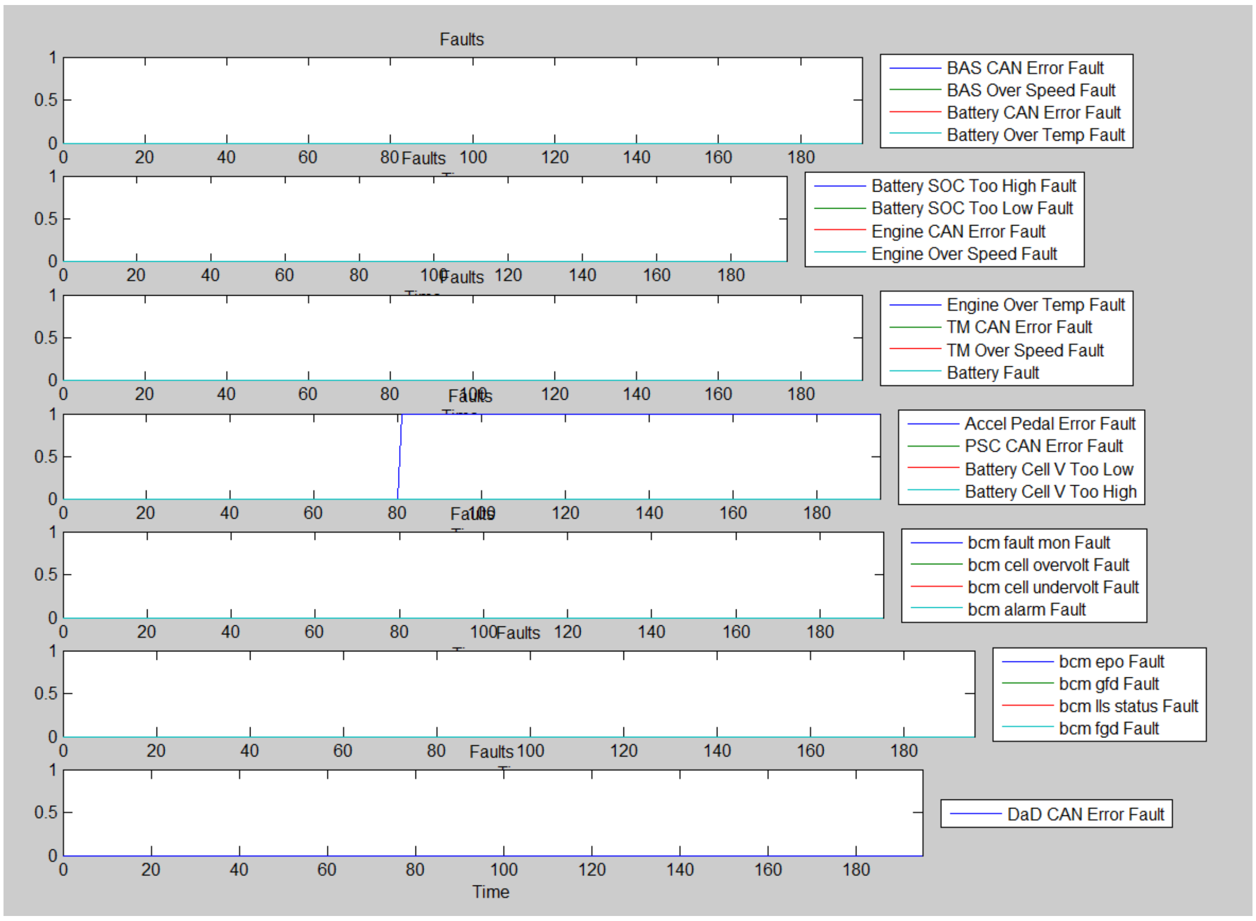


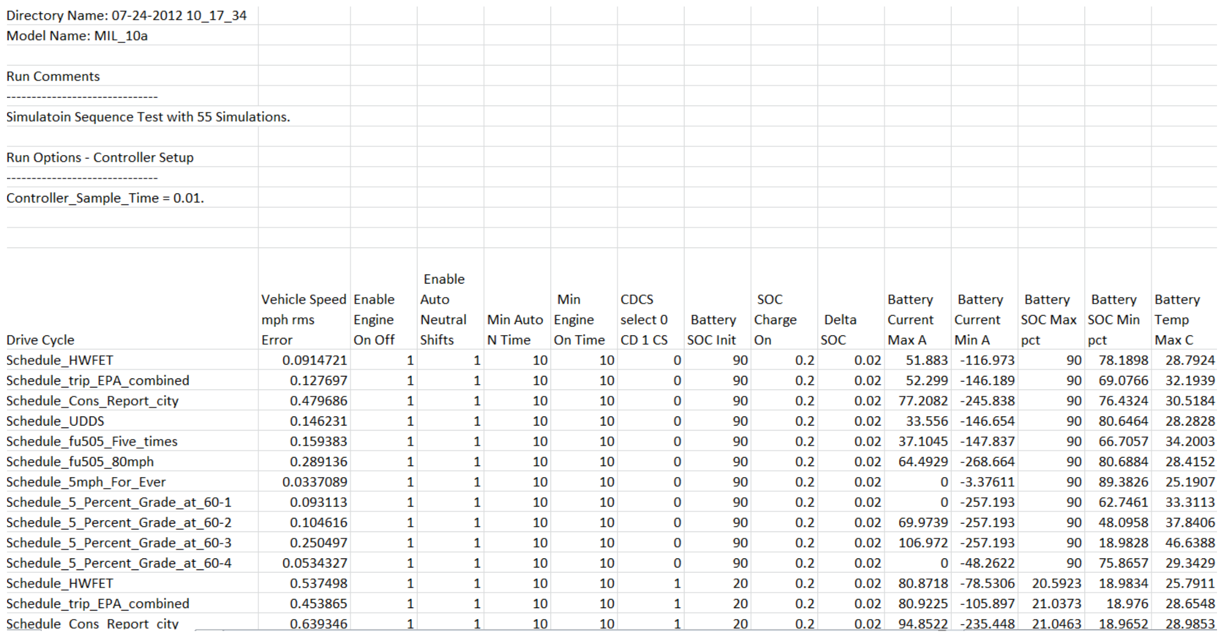
Figure 3: Accelerator pedal fault.

The script also generates a summary of off the runs stating how many faults were inserted and how many faults were recognized. The summary output from the script is shown below.

```
Directory Name: 07-13-2012 18_50_02
Model Name: MIL_10a

Run Comments
------------------------------
Script test 6.

Run Options - Controller Setup
------------------------------
Controller_Sample_Time = 0.01.
Auto-N Min time Trans is required to be in D before Auto_N is engaged =
10 (s).

Number of faults tested = 30.
Number of faults detected = 30.
Script Run Time = 2200.23 Seconds.
```

Note that 30 faults were inserted and 30 faults were recognized. Also, 60 plots similar to the two above were generated, 2 plots for each fault inserted. This data is not included in this report.

- Simulation Sequence Drive Cycle Script runs multiple drive cycles with tunable parameters that allow RHIT to test the model over thousands of conditions. Presently 11 different drive cycles are used with 8 tunable parameters: Engine On/Off Enable, Auto-Neutral On/Off, Min Auto-N Time, Min Engine on time, CD/CS select, Initial Battery SOC, SOC Charge On, Delta SOC. Parameters can easily be added. The script examines the output and looks for various parameters that may indicate bad control behavior, including max battery discharge current, max battery charge current, Max and Min battery SOC, Max component temperatures, max torque slew rates, and max speeds. The script exports the results to an excel file as well as saves all of the plot outputs. A portion of the excel file is shown below:



Directory Name: 07-24-2012 10_17_34
Model Name: MIL_10a

Run Comments

Simulatoin Sequence Test with 55 Simulations.

Run Options - Controller Setup

Controller_Sample_Time = 0.01.

Drive Cycle	Vehicle Speed mph rms Error	Enable Engine On Off	Enable Auto Neutral Shifts	Min Auto N Time	Min Engine On Time	CDCS select 0 CD 1 CS	Battery SOC Init	SOC Charge On	Delta SOC	Battery Current Max A	Battery Current Min A	Battery SOC Max pct	Battery SOC Min pct	Battery Temp Max C
Schedule_HWFET	0.0914721	1	1	10	10	0	90	0.2	0.02	51.883	-116.973	90	78.1898	28.7924
Schedule_trip_EPA_combined	0.127697	1	1	10	10	0	90	0.2	0.02	52.299	-146.189	90	69.0766	32.1939
Schedule_Cons_Report_city	0.479686	1	1	10	10	0	90	0.2	0.02	77.2082	-245.838	90	76.4324	30.5184
Schedule_UDDS	0.146231	1	1	10	10	0	90	0.2	0.02	33.556	-146.654	90	80.6464	28.2828
Schedule_fu505_Five_times	0.159383	1	1	10	10	0	90	0.2	0.02	37.1045	-147.837	90	66.7057	34.2003
Schedule_fu505_80mph	0.289136	1	1	10	10	0	90	0.2	0.02	64.4929	-268.664	90	80.6884	28.4152
Schedule_5mph_For_Ever	0.0337089	1	1	10	10	0	90	0.2	0.02	0	-3.37611	90	89.3826	25.1907
Schedule_5_Percent_Grade_at_60-1	0.093113	1	1	10	10	0	90	0.2	0.02	0	-257.193	90	62.7461	33.3113
Schedule_5_Percent_Grade_at_60-2	0.104616	1	1	10	10	0	90	0.2	0.02	69.9739	-257.193	90	48.0958	37.8406
Schedule_5_Percent_Grade_at_60-3	0.250497	1	1	10	10	0	90	0.2	0.02	106.972	-257.193	90	18.9828	46.6388
Schedule_5_Percent_Grade_at_60-4	0.0534327	1	1	10	10	0	90	0.2	0.02	0	-48.2622	90	75.8657	29.3429
Schedule_HWFET	0.537498	1	1	10	10	1	20	0.2	0.02	80.8718	-78.5306	20.5923	18.9834	25.7911
Schedule_trip_EPA_combined	0.453865	1	1	10	10	1	20	0.2	0.02	80.9225	-105.897	21.0373	18.976	28.6548
Schedule_Cons_Report_city	0.639346	1	1	10	10	1	20	0.2	0.02	94.8522	-235.448	21.0463	18.9652	28.9853

The script takes advantage of the MathWorks parallel processing toolbox and will allow RHIT to exercise the model through potentially thousands of combinations of parameter and drive cycle combinations. Note that the Excel file can be formatted to identify any values that are out of bounds and flag the simulation.

The RHIT fault strategy encompasses four areas. (1) Each controller added by RHIT (HVSC, PSC, DaD) runs a logic loop in parallel with a fault detection loop. The logic loop is responsible for implementing the functions required of that controller, and is unique to that controller. The fault detection in each controller is nearly the same and is responsible for detecting faults throughout the vehicle. Thus, we have software-parallel logic and fault detection functionality. (2) Since RHIT is adding three controllers, all capable of fault detection, the vehicle has parallel hardware fault detection. Any of the three controllers can recognize faults and respond to the situation. (3) Most faults are recognized at the subcomponent level. In most cases, the subcomponent is responsible for recognizing a fault and reporting the condition to the RHIT control system. The RHIT control system will respond to the fault, and in some cases the subcomponent controller may act as well. For some faults, the RHIT controller will perform a calculation to determine a fault, such as an improper torque sum. (4) The fault mitigation strategy is very conservative. In most cases the offending component is disabled and removed from the HEV control strategy. In some cases that involve high voltage, all motors, the ESS, and the DC-DC are disabled. The general strategy is usually to shut down components in a sequence so that the vehicle can enter a limp-home mode. Typically, the sequence is (1) disable the offending component, (2) disable the ESS, (3) disable the vehicle. Attempts 1 and 2 allow a limp home mode on the remaining subcomponents, while shutting down the vehicle kills everything and strands the motorist.

In addition to hardware-parallel and software parallel fault detection, fault detection is implemented logically after the logic functions so that no matter what the logic of the controller (HVSC, PSC, DaD) requests, the fault mitigation strategy takes precedence. This is shown in Figure A where the structure of the controller is discussed. Finally, both soft-stop and hard-stop (EDS) panic buttons are provided. The soft-stop button signals the HVSC to initiate a graceful shutdown where all motor torques are set to zero and the ESS contactors are not opened until the battery current falls below a threshold. With a hard stop (EDS), the 12 V enable line to the ESS and fuel pump are physically interrupted to "shut down" the path to all stored energy sources.

Thus, the levels of protection are: (1) The control systems recognizes a fault and mitigates the fault. (2) The user shuts down the vehicle with a "soft-Stop." (3) the user shuts down the vehicle with the EDS.

RHIT handles all safety-critical scenarios similarly by first creating a Fault Tree Analysis. This will allow RHIT to determine the root causes that will lead to the scenario. RHIT will then add components in the model to simulate each cause of the fault and add to the model to recognize this scenario as a potential lead to a fault. Then, either find a solution or set the corresponding action to the problem.

An example of a safety-critical scenario would be when the Battery experiences an Over Temperature fault as shown in Figure 4:

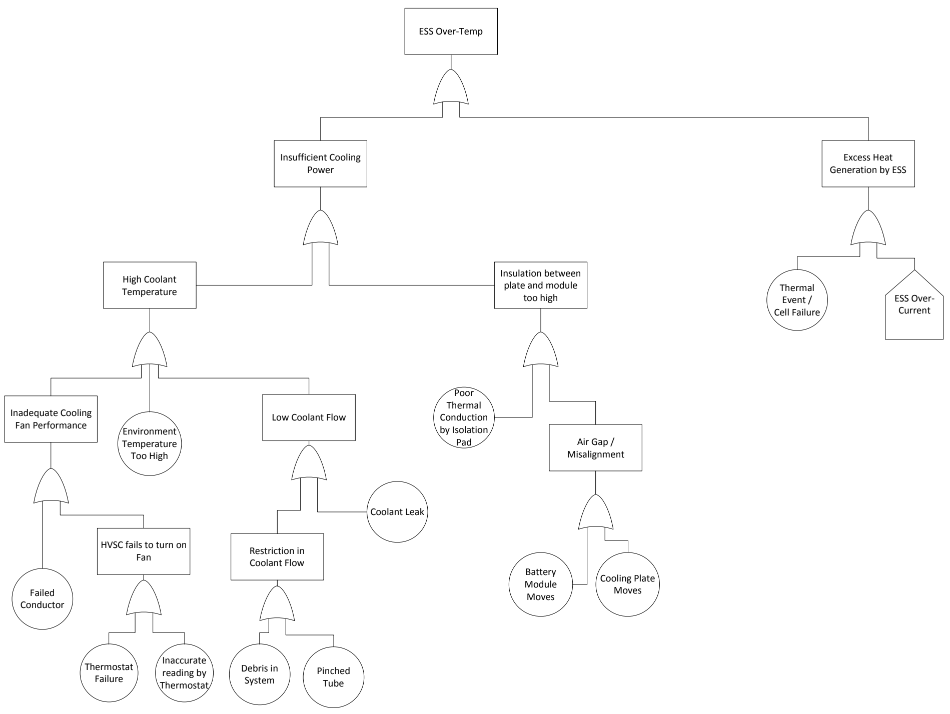

ESS Over-Temp
Insufficient Cooling Power
Excess Heat Generation by ESS
High Coolant Temperature
Insulation between plate and module too high
Thermal Event / Cell Failure
ESS Over-Current
Inadequate Cooling Fan Performance
Environment Temperature Too High
Low Coolant Flow
Poor Thermal Conduction by Isolation Pad
Air Gap / Misalignment
Coolant Leak
HVSC fails to turn on Fan
Restriction in Coolant Flow
Battery Module Moves
Cooling Plate Moves
Failed Conductor
Thermostat Failure
Inaccurate reading by Thermostat
Debris in System
Pinched Tube

Figure 4 - ESS Over-temp

In this case, the root causes can be a failed conductor, or a thermostat failure, or debris in the coolant system, and etc. In our model simulation we will simulate how these root causes will cause the temperature to rise. The model will monitor the temperature constantly as it is running and at a certain threshold value (this value is what we deem the point where the situation is potentially safety critical) will take actions necessary, such as; opening the battery contactor and disabling the battery from use.

4 Control System Hardware

RHIT will be using a distributed control system comprised of three MotoTron controllers to implement the various functions of the HEV strategy, control or communicate with specific components, and run fault detection and mitigation. The three controllers are:

- HVSC (Hybrid Vehicle Supervisory Controller – This controller runs all of the HEV torque blending algorithms, runs a fault detection algorithm, and is the overall supervisor of the system. If is realized with a MotoHawk ECM-5554-112 Pin Target running at and is physically located in the rear of vehicle.
- PSC (Powertrain System Controller) – This controller resides in the engine compartment and controls and analog signals needed by the local components including the engine, transmission, and BAS motor. For example, if analog signals are needed for the engine throttle, they will be generated by the PSC. If the accelerator pedal requires analog signals to be read, and the wiring is closest to the PSC, the PSC will read the pedal signals and perform the accelerator pedal fault detection. In year three, the PSC will also run the same fault detection algorithms as the HVSC so that redundant fault detection can be implemented. This controller will be implemented on a MotoHawk ECU555-80 Pin Target running at 40 MHz.
- DaD (Dashboard Display) – This controller will reside in the driver compartment under the dashboard and is responsible for reading switches, sensors, and signals in driver compartment, and interfacing with the driver GUI. In year three, the PSC will also run the same fault detection algorithms as the HVSC so that triple-redundant fault detection can be implemented. This controller is deployed on a MotoHawk ECM-0555-048-0403-C running at 40 MHz.

In year three, a parallel fault detection system will be implemented that includes a voting algorithm. For safety-critical faults such as an unintended acceleration, if one of the controllers detects a fault, the vehicle will be shut down immediately. For lower priority faults, such as an engine over-temperature, at least two of the controllers would have to detect the fault in order for corrective action to be taken. These methods are still fluid. However, the three controllers do provide the opportunity for a very creative and effective fault detection system.

A major reason for implementing distributed control system, aside from redundant fault detection, is the elimination of long wire runs. RHIT will be adding several components throughout the vehicle, each requiring analog and CAN communication signals. If a single controller was used to control all components, the addition of long analog wires runs would be required throughout the vehicle. This would greatly reduce reliability and flexibility. With three controllers, each located in a different compartment in the vehicle, making changes and additions is greatly simplified. Adding a sensor or cooling fan will be a simple task, rather than a cumbersome wire run that requires the addition of a wire to an already packed conduit that snakes from the front of the vehicle to the back.

A communication topology diagram is provided below in Figure 5. This diagram shows the major control modules in the vehicle, and the communications connections that exist between them. The HVSC, PSC, and DaD are shown in light blue with rounded corners, while the other component controllers are given as gray squares. This diagram is by no means exhaustive in terms of the final control system integration. Components such as the BCM, EBCM (Electronic Brake Control Module), EPSM (Electric Power Steering Module), and the GMLAN are not present. The focus of this diagram is on the major control components that will be integrated by Rose-Hulman, and the core interfaces between them such as the CAN and component enable lines.

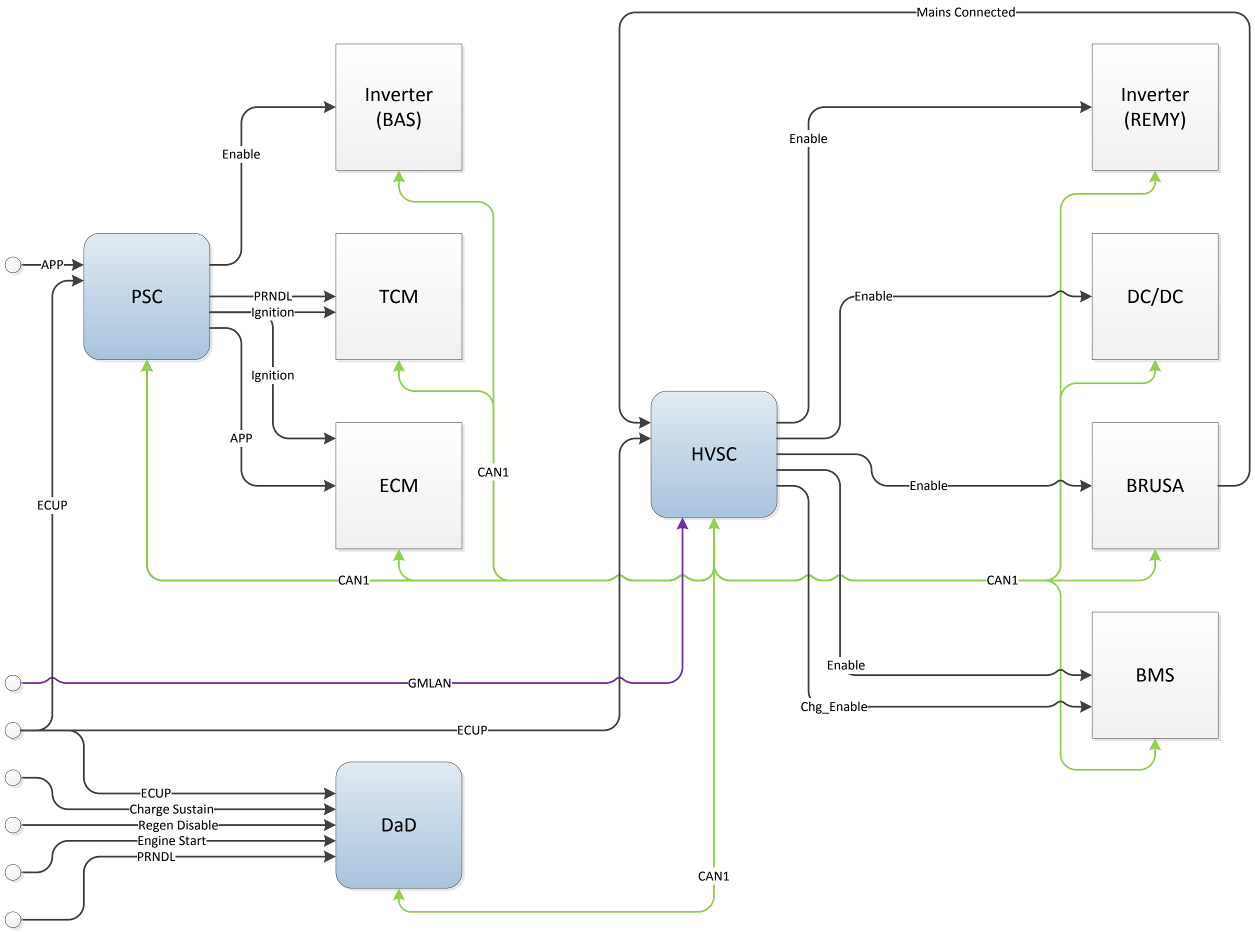


Figure 5 - Control System Block Diagram

Within the controllers themselves, subdivisions exist to help maintain control over on-going development, and to ease transfer of the models and controllers between the different development stages of MIL, SIL, HIL, and VIL. The major subdivisions consist of separating the interface portion from the actual functional logic. Since the interface varies from each stage, such as MIL and SIL making use of pass-through and HIL requiring hardware I/O, these blocks may be modified for each stage of development. Being able to modify these subsystems without touching the functional logic helps ensure that unwitting changes to the actual logic are not made so that the same logic is tested at each stage. It also helps to speed the process of moving between stages, as functional blocks can simply be "moved" between different sets of existing interface blocks. The implementation of this approach is given below in Figure 6, where the functional logic block is shown between the input and output interface blocks.

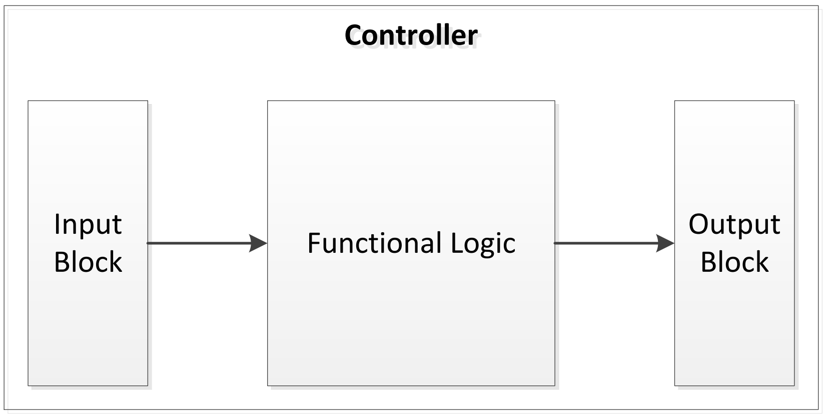


Figure 6 - Controller Subdivision

The top level logic layer of the controller consists of two parallel subsystems, one for controlling the operation of the vehicle (start-up, torque blending, shifting, charging, etc.) and one the monitors faults and disabling or shutting down the vehicle. A block diagram is shown in Figure 7:

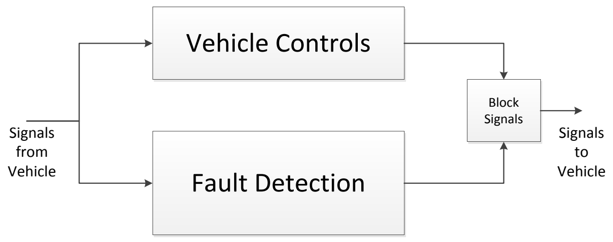
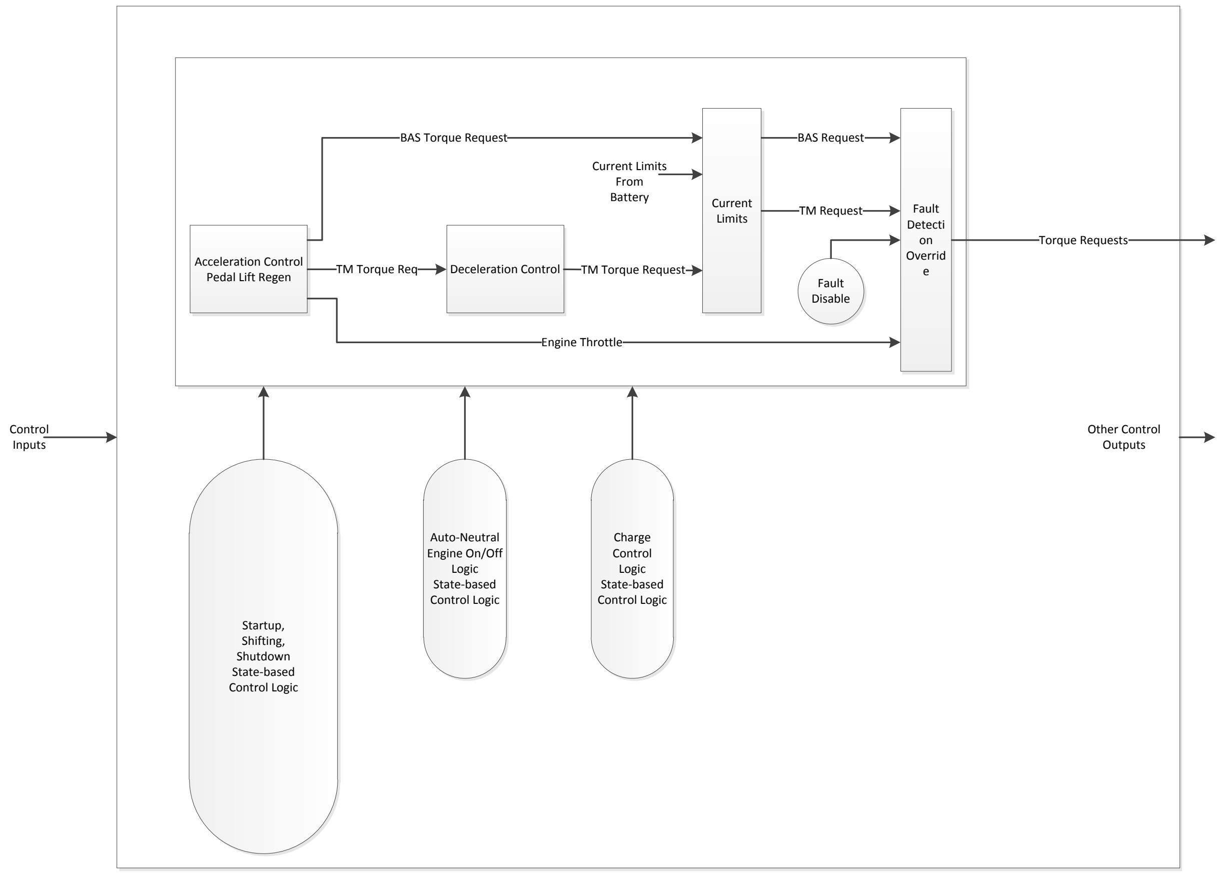


Figure 7 - Parallel Fault Detection

With this structure, the Fault Detection block takes precedence over the vehicle controls block through the Block Signals block. When no fault is detected, this block is a direct pass-through, allowing the vehicle control signals to pass through unchanged. If a fault is detected, the vehicle control signals are overridden, and the signals for a component are typically set to zero and a shut-down sequence for that component or components is initiated. When control logic is changed, all modifications are made within the Vehicle Controls block. This way, no control change can affect the Fault Detection's block ability to disable a component.

The Fault Detection block is one large State-Based logic block that looks at all signals and determines if a fault has occurred. In most cases, faults are simple bounds checks like over-temperature or over-speed. In some cases, a small amount of logic is used to detect a fault, such as a broken shaft where two speeds are not consistent. More complex fault detection is used for unintended accelerations where a torque sum is used. The output of this block usually shuts down a component by zeroing the torque request and disabling the component. For motors and the battery, the battery is shut down gracefully by setting all torques to zero and allowing the battery current to decay to below 5 amps before the contactors are opened. (Note that if the graceful shut-down is not obtained, the contactors are opened under load.)

A top level block diagram of the logic portion of the HVSC is shown in Figure 8, and shows the major HEV functions of the HVSC.



BAS Torque Request
Current Limits From Battery
Current Limits
BAS Request
TM Request
Fault Detection Override
Acceleration Control Pedal Lift Regen
TM Torque Req
Deceleration Control
TM Torque Request
Fault Disable
Torque Requests
Engine Throttle
Control Inputs
Other Control Outputs
Startup, Shifting, Shutdown State-based Control Logic
Auto-Neutral Engine On/Off Logic State-based Control Logic
Charge Control Logic State-based Control Logic

Figure 8: HVSC level block diagram.

Acceleration Control and Deceleration Control realize the torque blending, regenerative braking, and pedal-lift regen functions of the controller. The flow of these two blocks suggests that deceleration control overrides acceleration control. Current limits override the Accel and Decel blocks, and fault protection overrides everything.

The State-Based Start-up, Shifting, and shutdown control logic implements the following functions:

- Start-up - Prohibits the vehicle from starting unless it is unless in park, the key is on, and the battery is not charging. Performs a component wakeup and status check. Requires the brake to be on in order to start the engine.
- Shifting – Enforces shifting rules: Foot on brake to shift out of park. Zero speed for certain shifts. APP must be zero in order to shift.
- Shutdown – Enforce a soft shutdown: set all torque requests to zero. Wait for battery current to go below a threshold before opening contactors. Disable all components.

The Auto-Neutral, Engine Auto On/Off logic enforces the rules for when the transmission can automatically shift into and out of Neutral, and the rules for when the engine can automatically turn on or off. The rules are listed below:

- Auto-Shift Into Neutral – If the transmission is in drive, and the vehicle is not charging from one of the motors, and Auto-Neutral shifts are enabled, and there are no system faults, and the driver APP is less than 50%. (When the APP is 50% or less, only the TM is used to move the vehicle.)
- Auto-Shift out of Neutral – If the driver APP is greater than 50%, or Auto-Neutral is disabled, or the vehicle shifts out of drive, or the battery needs to charge, or a fault occurs.
- Engine Auto On – Same as Auto-Shift out of Neutral with the added condition (logic OR) of Engine Auto On/Off is disabled.
- Engine Auto Off – Same as Auto-Shift Into Neutral with the added condition (logic AND) that Engine Auto On/Off is enabled.
- The logic is temporal:
    - A sequence of events is required.
        - The transmission must auto shift to neutral

before the engine turns off.
    - The engine must turn on before the transmission auto-shifts to drive.
- The engine can turn on immediately. There is no minimum off time.
- The engine must be on for a minimum amount of time (which is tunable) before the engine is allowed to turn off.
- The transmission can auto shift into drive immediately. There is no minimum time the transmission must spend in neutral.
- The transmission must be in Drive for a minimum amount of time (which is tunable) before it is allowed to auto-shift into neutral.

Figure 9 shows a CD-mode simulation of a repeated FU505 cycle, and demonstrates the engine auto On/Off operation:

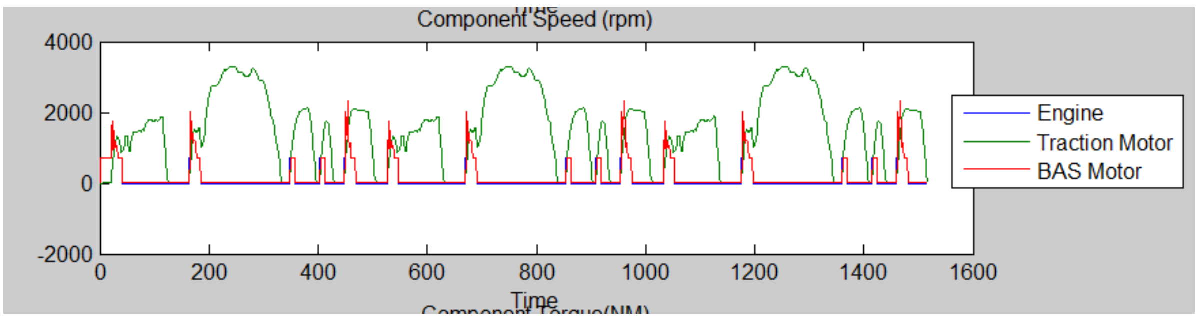


Figure 9: CD-mode simulation of a repeated FU505 cycle.

Although the torque plot is not shown, it can be inferred from the green trace which follows the drive cycle (FU505 3 times). Speed and torque traces are shown in our script plots, but not shown here as the plot sizes would be too small. When a high torque is required, say when the vehicle speed starts at zero speed and accelerates rapidly, the engine turns on and provides traction torque. When the vehicle requires less torque, the engine turns off. Figure 10 shows the torque for the Engine, TM, and BAS:

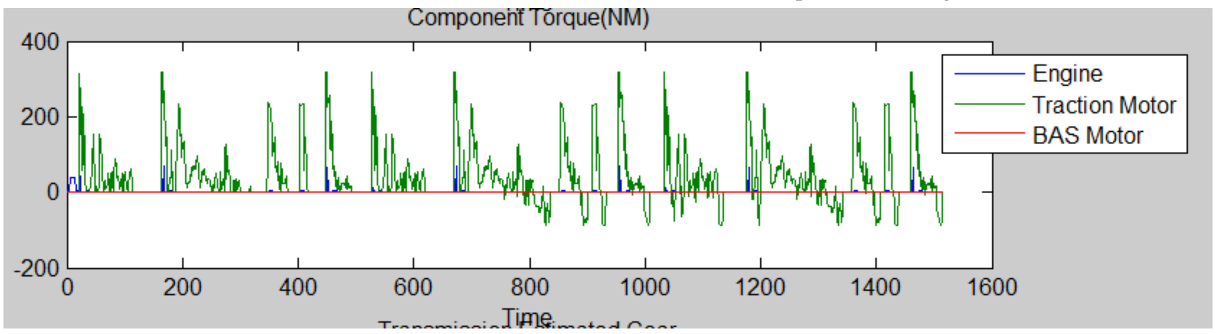


Figure 10: Vehicle torque plots for the Engine, TM, and BAS

The plot shows that the engine only comes on during times of high torque demand, and then turns off after 10 seconds of low torque demand. This plot verifies both the torque blending in CD mode and Auto-Neutral and Engine Auto-On/Off operation. Figure 11 shows an enlarged view of the component speeds showing that the engine must remain on for 10 seconds (min on time setting for this simulation) before the engine turns off:

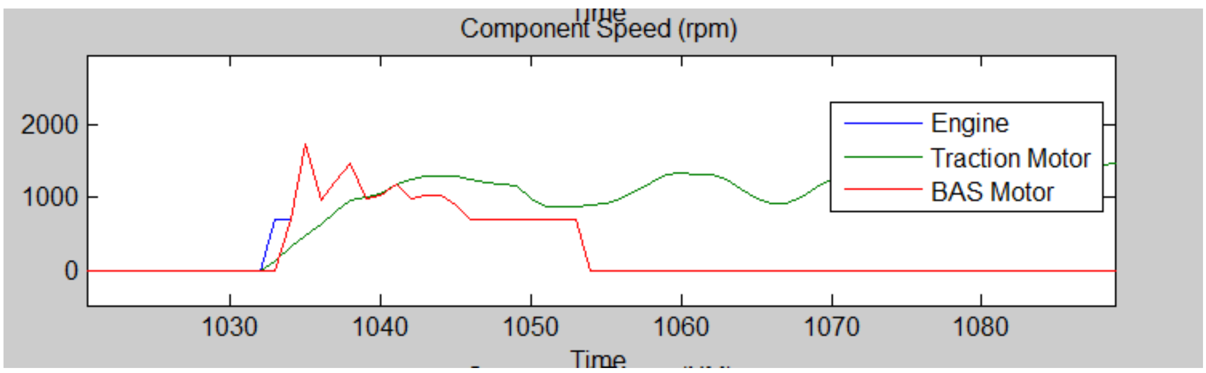


Figure 11: Zoom of vehicle torque plots for the Engine, TM, and BAS

Note that the engine and BAS speeds are mostly the same for all of the plots.

The control algorithms in CS mode are somewhat primitive at this point. The control system does hold the SOC within the specified SOC band, however, the algorithm is more focused on successfully operating a vehicle rather than operating a vehicle efficiently. (Just lay it on the line! A vehicle that works garners more points than one that does not…) Figure 12 shows the vehicle in CS mode running the US06 Highway cycle three times is shown below:

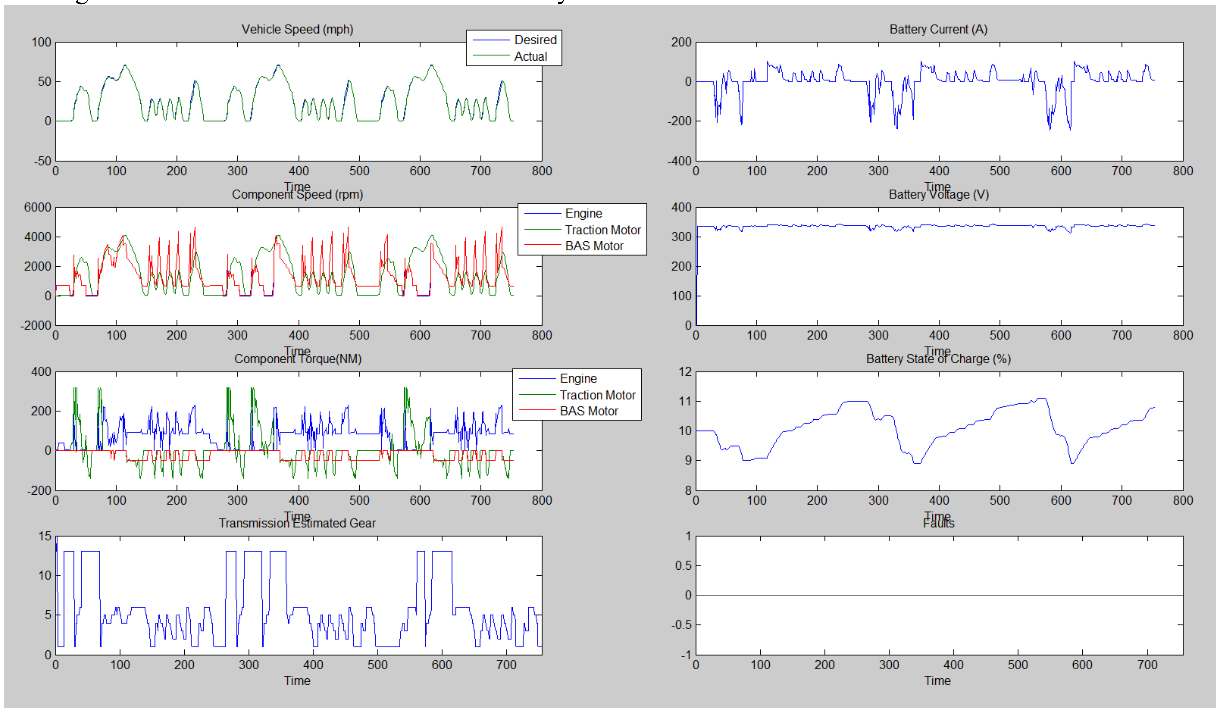


Figure 12: Simulation of the vehicle in CS mode running the US06 Highway cycle three times.

The plot verifies that the algorithm can keep the battery within a specified band. A narrow band and long drive cycle was chosen so that the SOC variance could be observed.

Next, we will look at a torque plot. The torque strategy in CS mode is to take advantage of the strategy in CD mode that has already been developed. When the battery SOC is high enough, the CD algorithm will be used where the TM is the main tractive source, the engine is only used during times of high driver demand, and Auto-Neutral and Engine Auto On/Off are available. When the SOC hits a lower threshold and battery charging is needed, the engine becomes the main tractive source and both motors can be used to charge the battery. Presently, for battery charging, the BAS motor is used at all speeds and the TM is used at high speeds. However, this blending can be changed in future improvements. Since the battery has low SOC and cannot provide tractive power, the engine must provide all of the tractive torque, even during times of high torque demand. To meet these needs, if a high torque request is received while the battery is charging, the charging request is reduced so that more of the engine torque can be applied to honoring the torque request.

To illustrate the various modes of operation, we will zoom in on the torque plot at various points in the cycle. Figure 13 shows the torque when the battery SOC is high enough so that charging is not required:

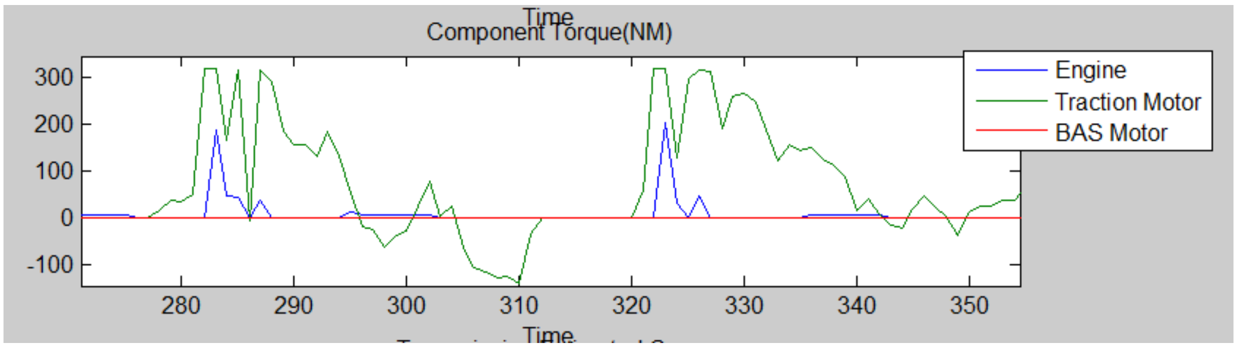


Figure 13: Vehicle torque plots when battery SOC is high.

We see that the BAS motor supplies no torque, the TM supplies the majority of the tractive torque and the engine is only used during times of high torque demand. This is the same as in CD mode.

Figure 14 shows the torques when the battery is charging. The plot verifies that the BAS motor charges at all vehicle speeds, the TM only charges at high speeds, and that charging is reduced or set to zero at times of high torque demand:

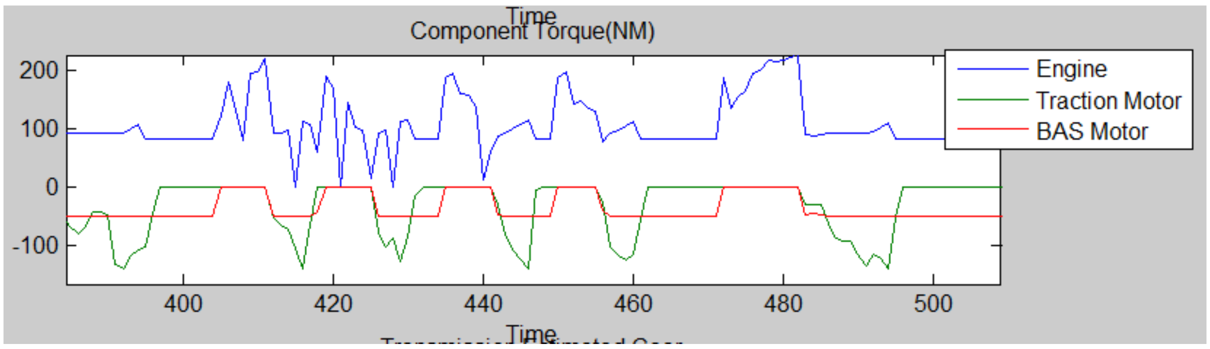


Figure 14: Vehicle torque plots when battery is charging.

As safety is RHIT's number one priority, the team ensures that whenever making a design choice safety is never a trade-off. Currently, the RHIT controller monitors over 20 vehicle diagnostics and constantly ensures that the vehicle is operating without harm. If, however, a fault could potentially occur, the controller is designed to detect, recognize and control the situation before it gets out of hand. This is particularly important in the more safety-critical scenarios such as an Unintended Acceleration and a vehicle component experiencing an Over-Temperature.

Figure 15 and Figure 16 show the Fault Tree Analysis and simulation plots of the event of an Unintended Acceleration. It is one of a number of safety-critical scenarios the RHIT controller can handle. RHIT utilizes Fault Trees to be able to analyse a larger scope for the possibility of a fault. Then, after adding the root causes as possible simulation faults in the model, the controller is tested in real time simulation to see if it is responding and handling the problem as it should and if not, adjusted accordingly. Development will always start with a simple solution: by shutting down the faulty component and then operating the vehicle in a limp-home mode if possible.

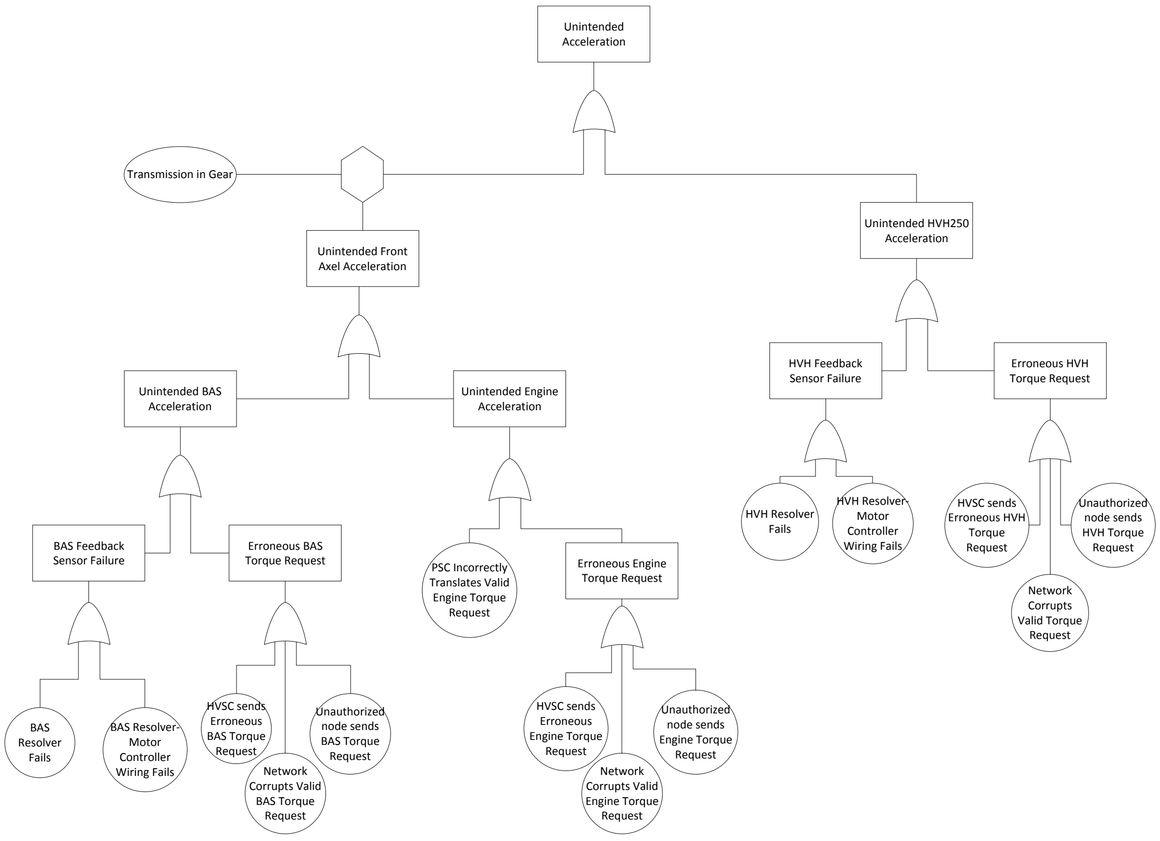

Unintended Acceleration
Transmission in Gear
Unintended Front Axel Acceleration
Unintended HVH250 Acceleration
Unintended BAS Acceleration
Unintended Engine Acceleration
HVH Feedback Sensor Failure
Erroneous HVH Torque Request
HVH Resolver Fails
HVH Resolver-Motor Controller Wiring Fails
HVSC sends Erroneous HVH Torque Request
Unauthorized node sends HVH Torque Request
Network Corrupts Valid Torque Request
BAS Feedback Sensor Failure
Erroneous BAS Torque Request
PSC Incorrectly Translates Valid Engine Torque Request
Erroneous Engine Torque Request
BAS Resolver Fails
BAS Resolver-Motor Controller Wiring Fails
HVSC sends Erroneous BAS Torque Request
Unauthorized node sends BAS Torque Request
Network Corrupts Valid BAS Torque Request
HVSC sends Erroneous Engine Torque Request
Unauthorized node sends Engine Torque Request
Network Corrupts Valid Engine Torque Request

Figure 15 - Unintended Acceleration

The two plots below show two faults: a Traction Motor and an Engine Unintended Acceleration (as is presented below). Each time after a fault is introduced, detected by the large spike in torque, the plot shows how the controller struggles with the problem for a short period of time as it diagnoses what the vehicle is undergoing. After it recognizes that it is a safety-critical scenario, the controller disables the harmful component and has a working component take over. In this case, the Engine first takes over the disabled Traction Motor and then is disabled itself as it also recognizes that the Engine too experiences an unintended acceleration.

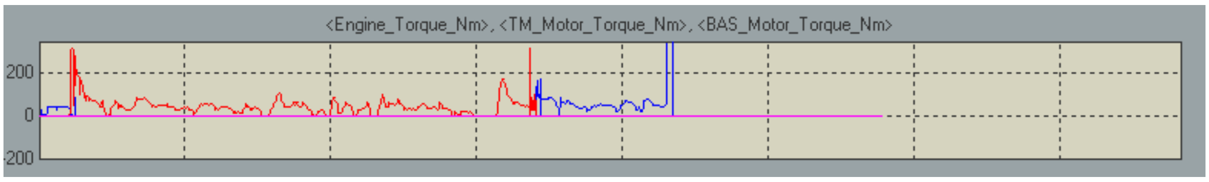


Figure 16 - Vehicle torque plots during an unintended acceleration fault.

After the Engine experiences an unintended acceleration, notice that the vehicle is no longer following the designated drive cycle. This is because the Engine and both motors are disabled due to the unintended acceleration it previously experienced. Currently, the controller will shut down the component that experiences and unintended acceleration. This could potentially be a problem if the vehicle experiences both a Traction Motor and Engine unintended acceleration at the same time. However, as mentioned previously, RHIT is currently not ready to handle on-the-spot situation and the present solution seems appropriate.

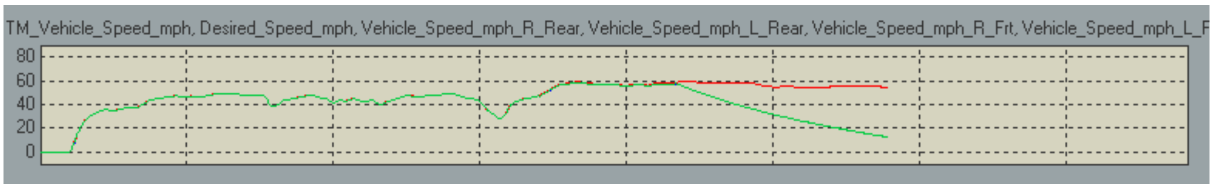


Figure 17 - Vehicle speed during an unintended acceleration fault.

System Safety comes in two different forms. The largest component is the fault detection subsystem in each controller (HVSC, PSC, DaD) that monitor fault codes and calculate faults. This is a state-flow chart that keeps tract of faults and shuts down components when necessary. Another level of safety are “analog” subsystems within the controllers that prevent certain problems for occurring. Examples are torque slew limits, current limits, vehicle speed limits, zero APP when the brake is depressed, and fault detection on the APP. As an example, we will show the logic for reading the APP and generating faults if necessary. And, just to be annoying, we show it as a Simulink diagram.

The accelerator pedal has two potentiometers, pot1 and pot2. Five possible faults are possible: Pot1 high, Pot1 low, Pot2 high, Pot 2 low, and Pot1 and Pot 2 are inconsistent. Signals for each pot are calculated within a subsystem. The outputs of these subsystems are shown in Figure 18:

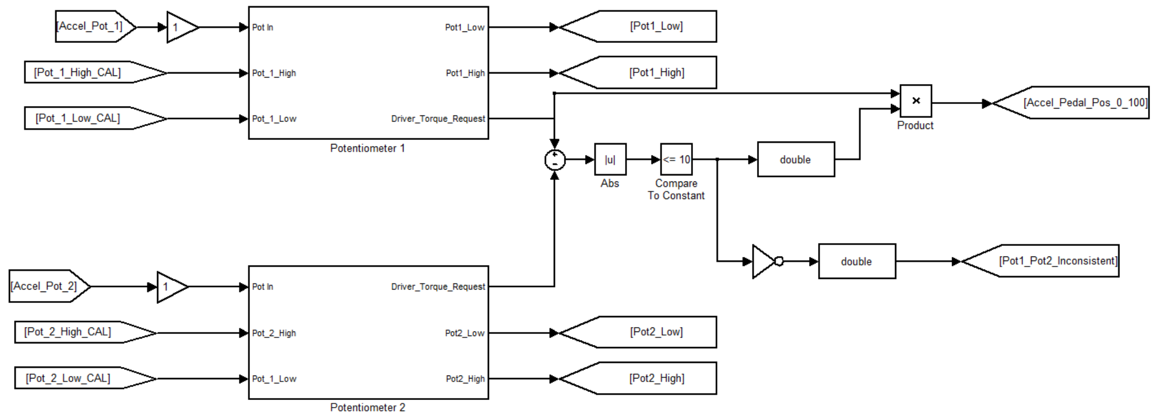


Figure 18: Accelerator pedal fault detection model.

Note that each subsystem emits a Driver_Torque_request signal which is the pedal position indicated by the corresponding pot. If an individual pot has a fault, the Driver_Torque_Request will be zero. However, if the pots individually do not have an error, then the Simulink model shown above compares the two driver torque requests. If the two requests differ by more than 10%, the Accelerator Pedal Signal is set to zero and a fault is generated (Pot1_Pot2_Inconsistent). This fault will be detected by the Fault Detection systems and then vehicle will be shut down.

The model inside the Potentiometer subsystems is shown in Figure 19:

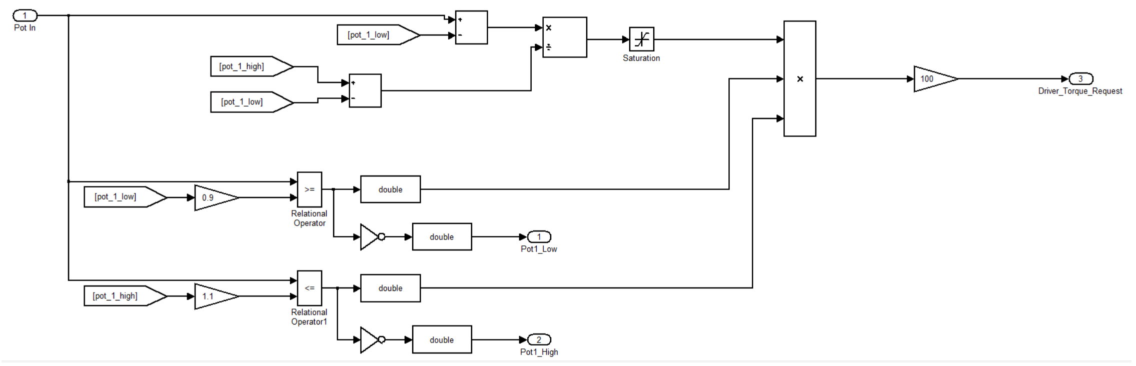


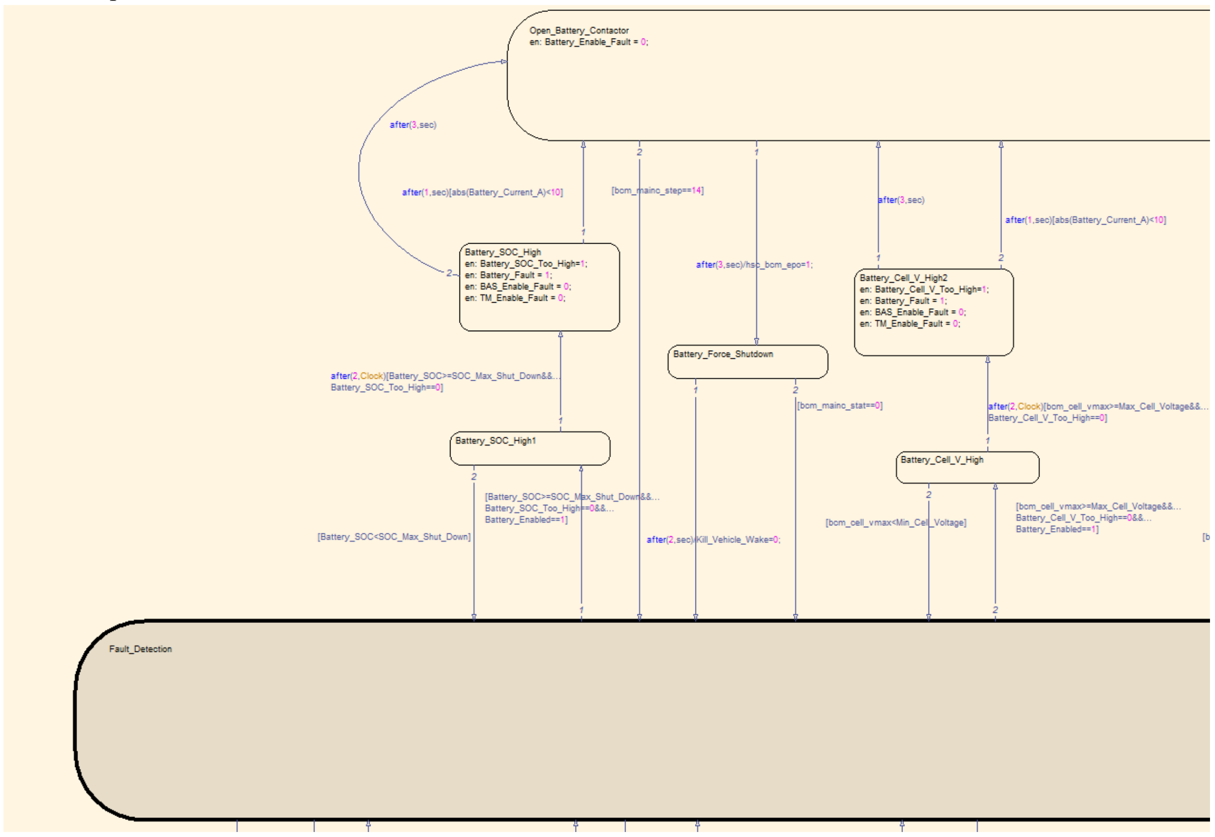
Figure 19: Accelerator pedal fault detection logic.

Each subsystem has two calibrations, pot_x_high and pot_x_low. If the value read from the pot is not within the high and low limits, a fault is generated (Pot1_Low or Pot1_High) and the Driver_Torque_Request is set to zero. If a fault is not detected, a Driver_torque_Request between 0 and 100 is output.

As mentioned above, several of the “analog” systems are used to implement various safety features. In addition to the analog safety, each controller has a fault detection system (or will as they will all run the same system). This system is implemented in Stateflow so that it can remember faults. A typical fault goes through the following sequence:

1) If the fault has already occurred, remember it, and don't react to it again. This is done to store faults as well as not waist processor time responding to a fault that has already been mitigated. (Assuming that the offending component has been shut down and that we have not re-enabled the component (which is never done on this system)).
2) If the fault is detected, make sure it is persistent. The fault must be active for a specified amount of time. This is done to avoid shutting down components due to an erroneous signal.
3) Shut down the component.
4) For some faults, shutdown the battery gracefully as well.
5) If the battery does not shutdown gracefully, force open the contactors.
6) Wait for the next fault.

This logic is implemented with Stateflow charts shown in Figure 20:



Open_Battery_Contactor
en: Battery_Enable_Fault = 0;
after(3,sec)
after(1,sec)[abs(Battery_Current_A)<10]
[bcm_mainc_step==14]
Battery_SOC_High
en: Battery_SOC_Too_High=1;
en: Battery_Fault = 1;
en: BAS_Enable_Fault = 0;
en: TM_Enable_Fault = 0;
after(3,sec)/hsc_bcm_epo=1;
after(3,sec)
after(1,sec)[abs(Battery_Current_A)<10]
Battery_Cell_V_High2
en: Battery_Cell_V_Too_High=1;
en: Battery_Fault = 1;
en: BAS_Enable_Fault = 0;
en: TM_Enable_Fault = 0;
Battery_Force_Shutdown
after(2,Clock)[Battery_SOC>=SOC_Max_Shut_Down&&...
Battery_SOC_Too_High==0]
[bcm_mainc_stat==0]
after(2,Clock)[bcm_cell_vmax>=Max_Cell_Voltage&&...
Battery_Cell_V_Too_High==0]
Battery_SOC_High1
Battery_Cell_V_High
[Battery_SOC>=SOC_Max_Shut_Down&&...
Battery_SOC_Too_High==0&&...
Battery_Enabled==1]
[Battery_SOC<SOC_Max_Shut_Down]
after(2,sec)/Kill_Vehicle_Wake=0;
[bcm_cell_vmax<Min_Cell_Voltage]
[bcm_cell_vmax>=Max_Cell_Voltage&&...
Battery_Cell_V_Too_High==0&&...
Battery_Enabled==1]
Fault_Detection

Figure 20: Stateflow logic implementing the fault detection system.

The vehicle plot for an accelerator pedal fault is shown in Figure 21:

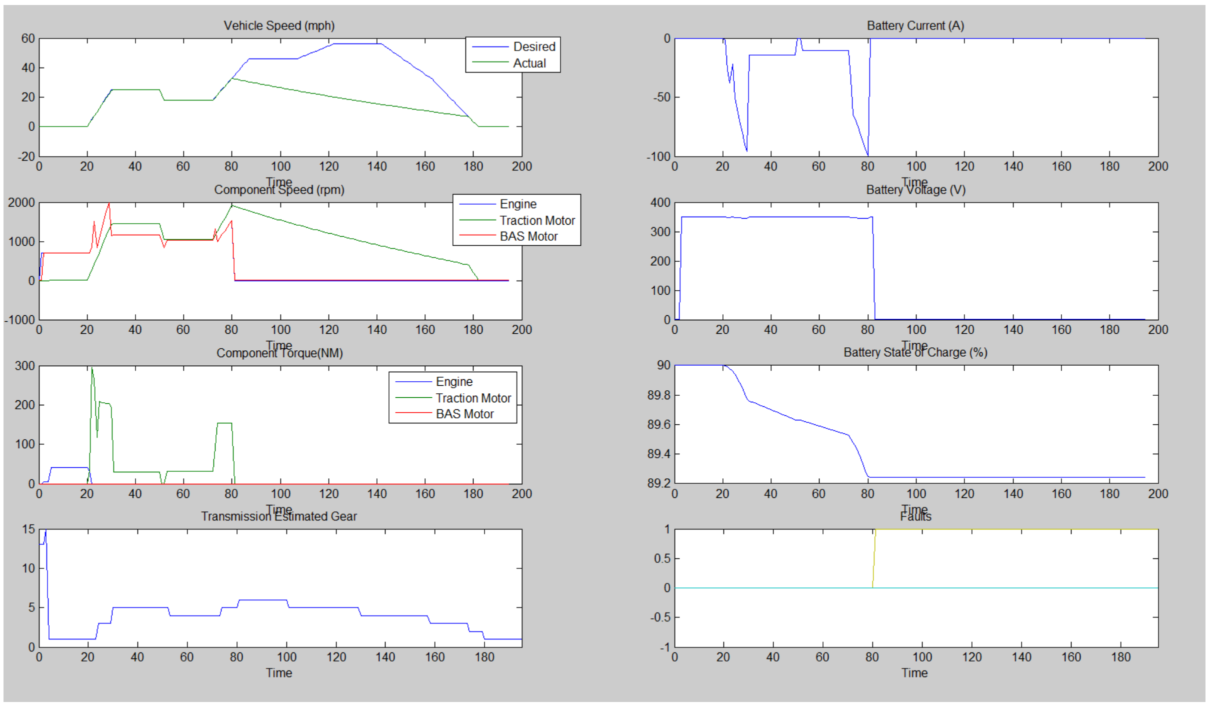


Figure 21: Vehicle plot for an accelerator pedal fault.

Note that all torques are set to zero, the battery is disabled, and then the vehicle slowly coasts to zero. A second fault plot is also generated, indicating what fault is recorded. In this case, the plots show an accelerator pedal fault. The plot is shown in Figure 22:

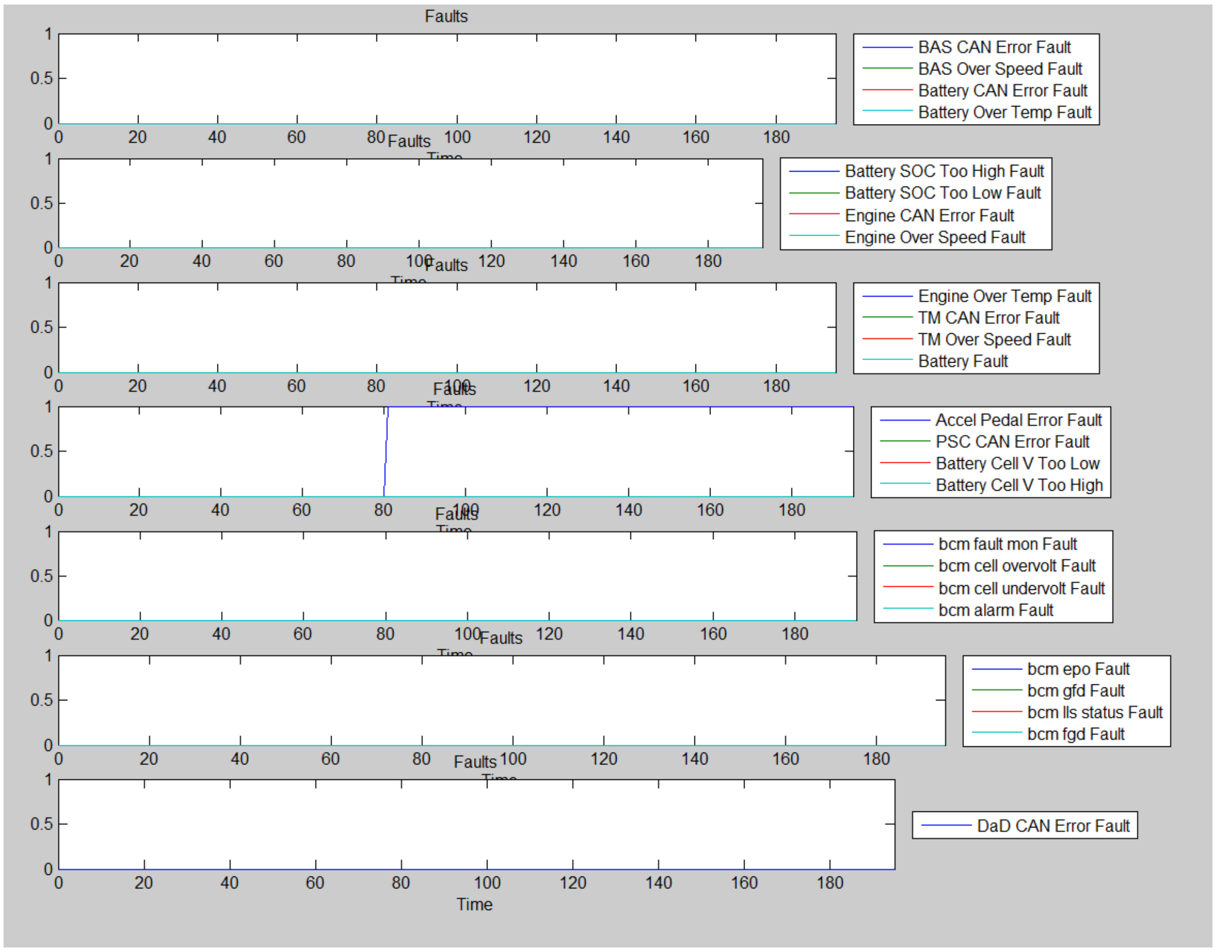


Figure 22: Fault plot for an accelerator pedal fault.

The RHIT controller is designed to be able to monitor the vehicle's diagnostics and verify that the vehicle is operating properly at all times. For all possible faults, RHIT will create a Fault Tree Analysis to determine root causes of each fault. The model then simulates the fault via each root cause. Through testing and development, the RHIT controller should be able to detect and recognize when the situation can turn into a fault and respond accordingly. For example, a cooling fan failure can lead to a rise in Engine temperature and lead to an engine over-temperature fault as shown in Figure 23. The controller, which is constantly monitoring the Engine's temperature, will detect that the temperature is rising. RHIT will give the controller a temperature safety threshold which, when passed, will allow the controller to respond before the engine actually overheats to avoid a safety-critical scenario.

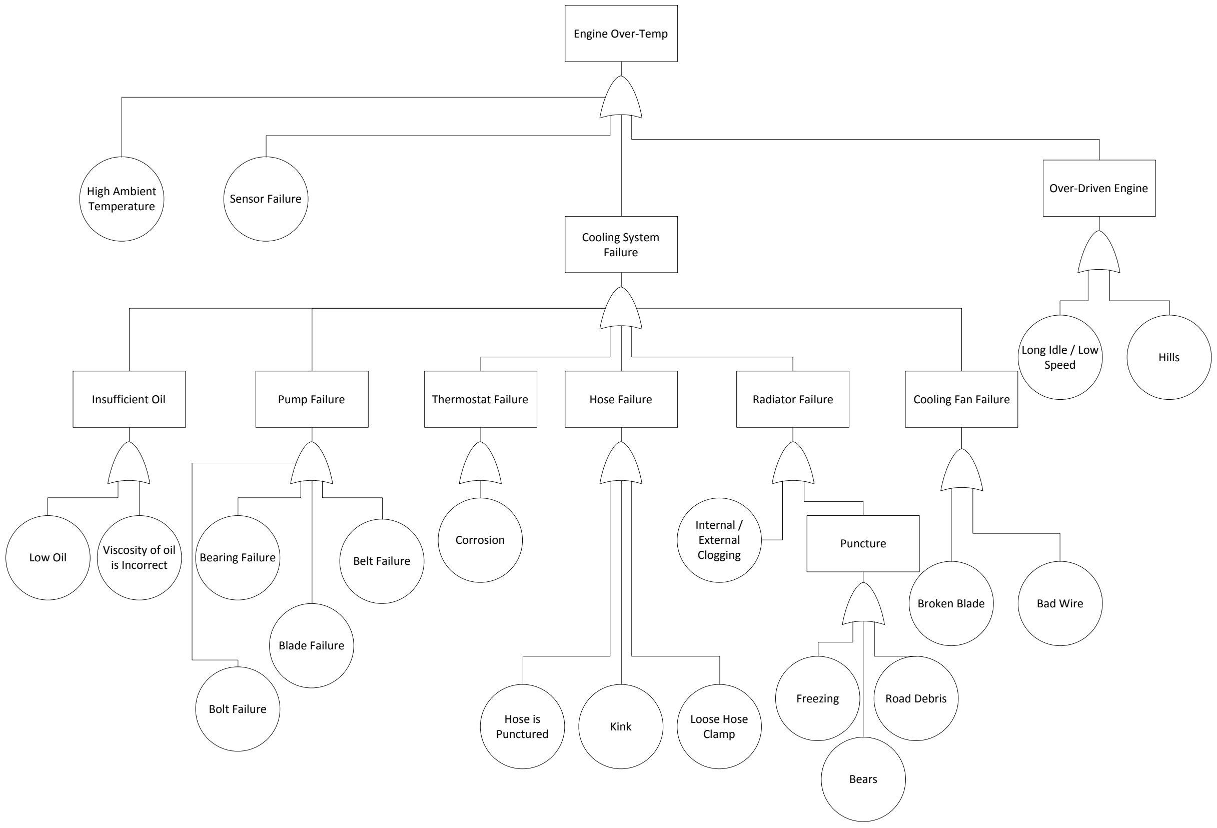


Figure 23: Fault tree analysis of an engine over-temperature fault.

RHIT exercises the Limp Home method. This allows the vehicle to still be operational when a component experiences a fault. In this situation, with the Engine shut off, the Traction Motor will take over and allow the driver to move to a safe location before analysing the situation.

Acknowledgments

The authors would like to acknowledge Kristen de la Rosa, EcoCAR2 Program Director, Argonne National Lab and Zach Pieri, EcoCAR2 Team Mentor, General Motors.

References

[1] *Rose-Hulman Institute of Technology,* http://www.rose-hulman.edu/, accessed on 25 June 2013

[2] EcoCAR2, http://www.ecocar2.org/ecocarchallenge, accessed on 25 June 2013.

[3] Duoba, M., Carlson, R., and Wu, J., "Test Procedure Development for "Blended Type" Plug-In Hybrid Vehicles," *SAE Int. J. Engines* 1(1):359-371, 2009, doi:10.4271/2008-01-0457.

[4] Woodward MotoHawk, http://mcs.woodward.com/, accessed on 25 June 2013.

[5] Li, X., Li, L., Sun, Y., Hu, Z. et al., "Optimization of Control Strategy for Engine

Start-stop in a Plug-in Series Hybrid Electric Vehicle," SAE Technical Paper 2010-01-2214, 2010, doi:10.4271/2010-01-2214.

Authors

	Marc Herniter is a Professor at Rose-Hulman Institute of Technology (Ph.D., Electrical Engineering, University of Michigan, Ann Arbor, 1989); Dr. Herniter's primary research interests are in the fields of modeling of complex systems, power electronics, hybrid vehicles, and alternative energy systems. He has worked on power electronic systems that range in power levels from 1500 W to 200 KW. He is the author of several text books on simulation software including PSpice, Multisim, and Matlab. He is currently the co-advisor for the Rose-Hulman EcoCAR Hybrid-Electric vehicle team and the faculty supervisor of Rose-Hulman's Model-Based-Systems Design laboratory.
	Zac Chambers has his BS and MS in Mechanical Engineering from Rose-Hulman Institute of Technology and his PhD in Engineering Science and Mechanics from the University of Tennessee, Knoxville. He has been on the ME faculty for Rose-Hulman since 2000 and has worked with Marc Herniter as Mechanical Advisor for ChallengeX, EcoCAR, and now EcoCAR2 racking up 9 years of experience in hybrid vehicle powertrain design and execution. He additionally serves as the Program Director for the Advanced Transportation System Program at Rose-Hulman.