An Overview of Hardware-In-the-Loop Testing Systems at Visteon

Syed Nabi and Mahesh Balike
Visteon Corporation

Jace Allen and Kevin Rzemien
dSPACE Inc.

Reprinted From: Testing and Instrumentation 2004
(SP-1871)
ABSTRACT

This paper discusses our experiences on the implementation and benefits of using the Hardware-In-the-Loop (HIL) systems for Powertrain control system software verification and validation. The Visteon HIL system integrated with several off-the-shelf diagnostics and calibration tools is briefly explained. Further, discussions on test automation sequence control and failure insertion are outlined. The capabilities and advantages of using HIL for unit level software testing, open loop and closed-loop system testing, fault insertion and test automation are described. HIL also facilitates Software and Hardware Interface validation testing with low-level driver and platform software. This paper attempts to show the experiences with and capabilities of these HIL systems.

INTRODUCTION

HIL technology has been in wide use in Defense and Aerospace industry as early as the 1950s. In spite of the high cost of the HIL technology in those days, the Defense and Aerospace industry took advantage of HIL systems mainly due to the risk of human life involved in the real time system and the extremely expensive prototype systems under test. The automotive industry did not embrace HIL technology for development and testing of automotive control systems mainly due to its high cost and feasibility of using the HIL technology for comparatively simpler control systems.

Over the years, the automotive industry started adding more and more diverse embedded software controlled electronic components in vehicles to improve performance; efficiency and comfort as well as to be compliant with government regulated and mandated emission requirements. The pressure of increased complexity in the design and verification process timing and cost involved with the testing of complex electronic and component systems forced the automotive industry to investigate and invest in better testing methodologies.

The advancement of computer technology, availability of highly efficient low cost compute engines and, readily available off-the-shelf systems, have pushed the viability and usability of the HIL technology from the Defense and Aerospace industry to the Automotive industry. After the successful use of HIL simulation technologies in the Aerospace industry, the Automotive industry adopted the technique in the 1990s. Several successful partial and complete closed loop simulations can be cited in the literature since then [1,2].

Figure 1 Traditional Development Process

The traditional design, development and verification process of automotive embedded control system software comprises several steps and iterations as depicted in Figure 1. As illustrated in the figure, during the design and verification process the design requirements may change due to internal process,
addition of new requirement, and refinement of requirement. The late changes in requirements cause a significant impact on the development and testing time of control system software and this iterative loop of requirement change, development and testing is not very efficient. Further, obtaining a prototype vehicle during the early software developmental phase is extremely difficult and cost prohibitive, especially for a non-OEM (Original Equipment Manufacturer) companies.

A Powertrain Control Module (PCM) is one of the most complex electronic embedded control systems in the modern vehicle. The PCM software therefore requires a rigorous and thorough testing of its functionality. In order to manage the increasing demand of technology in PCM design, Visteon Corporation started investigating the possibilities of using a state-of-the-art HIL system for PCM software development and verification in 1998. HIL technology is best suited to perform a complete and rigorous testing of the Embedded control system software of PCM. Visteon anticipated that HIL simulation would improve quality and also reduce development time and cost. At the same time the HIL system based development and testing will complement the implementation of model based software development (MBD) technology. Visteon anticipated that HIL simulation would improve quality and also reduce development time and cost. At the same time the HIL system based development and testing will complement the implementation of model based software development (MBD) technology. The initial strategy was to implement HIL for PCM software testing only. However, over time it became apparent that the same implementation could be easily tailored for the Software/Hardware Interface validation as well.

Implementation of a HIL system for testing during the development phase of PCM software also addresses other issues associated with the traditional bench test environment. Some of the issues associated with the traditional bench test environment are uncontrolled and non-repeatable tests without proper closed loop feedback to perform a complete system verification and validation of the software in real time, difficulty in achieving consistency, manual mode of operation for set points, ad hoc data capture and test results and the near impossible task of testing the complex closed loop control algorithms of the PCM software in an open loop bench test environment.

Visteon has proven that there are significant benefits of Open Loop HIL testing during the initial phase of software development by performing unit level software verification. HIL testing is also used for verification of the Software and Hardware Interface of the low-level platform software. The integration of HIL systems with calibration and diagnostic tools also provides significant advantages with closed loop simulation and testing of completely integrated software in real time on the bench.

Several other features of HIL such as test automation, test stimulus generation, and fault-insertion testing are addressed. These HIL systems use Python, an advanced test scripting language, to control test sequences and coordinate the variety of interfaces necessary for a particular test, or a set of tests. This test-scripting feature of the HIL system gives the test developer the capability to implement full lights-out testing scenarios and also allow for exact reproduction for subsequent DUT (Device Under Test) regression testing. The HIL systems also have the capability to inject stimulus signals into the DUT, which can be used for a variety of purposes. These include drive cycles for an engine or vehicle test, and also response data histories for component validation.

Finally, the interfaces between the PCM and the plant (Vehicle, Engine, etc) need to be tested. When failures occur with sensors or actuators, the PCM generally needs to detect these and provide for strategies to overcome them. Using the HIL systems, it is fully possible to simulate almost any type of failure that a controller could experience. This failure testing can also be performed during critical control periods, which might not be possible to implement on the actual target plant.

Visteon HIL System

ARCHITECTURE

The Visteon HIL software-testing environment is built on modified dSPACE midsize and/or full size engine simulators integrated with several off the shelf tools. The system architecture of the Visteon HIL system is shown in block diagram in Figure 2. The Visteon HIL simulator is housed inside a 19” desktop rack containing a Power PC processor board for real time computation, a specialized automotive HIL I/O board, additional I/O cards, required signal conditioning boards, load boards with fault simulation capability as the main component of the simulator. The system is also equipped with a
remote control power supply. The power supply can be controlled in real time during simulation, which provides the capability of vehicle battery simulation. The PCM is physically connected to the HIL simulator by customized harnesses for I/O routing either directly or through switchable breakout boxes. Further, it is possible to attach actual physical component assemblies to the HIL system using an external interface. The system also accommodates integration of off-the-shelf calibration and diagnostic tools. The integration of calibration and diagnostic tools with the HIL system provides the capability of automated testing of very complex Powertrain control system software. [3]

One of the critical parts of this HIL system is the load subsystem. It is very critical to have an appropriate load for the actuator signals in order for the PCM software to work properly. In order to accommodate the special characteristics of certain actuators, the system connects to a specially designed company proprietary load subsystem. The closed loop simulation of the system requires appropriate feedback response from the actuator load to satisfy the smart device driver integrated circuits (IC) and advanced On Board Diagnostics (OBD) system software. Without the appropriate closed loop system and actuator load feedback, the PCM controller will enter a fault mode and will defeat the purpose of validation of the system software. For the complete closed loop system simulation, another significant requirement is the representative plant model. The plant model should represent the physical behavior of the plant; in our implementation, the plant consists of the engine, the transmission, and other ancillary components.

A set of plant models suitable for real-time simulation has been developed and validated in-house, to represent various automotive subsystems for different powertrain applications. The real-time executable plant models are developed in MATLAB®, Simulink®, and Stateflow®, and are composed of flexible and configurable component and sub component models. Physical component and systems are represented in mathematical models. These representative plant models are developed with significant and adequate detail to perform the controller software algorithm verification and validation.

TEST AUTOMATION SEQUENCE CONTROL AND DESIGN

Phases of testing and automation

The task of powertrain hardware and software testing using HIL simulation requires many phases and interfaces. Refer to Figure 3 to see a depiction of some of these interfaces. Below is a list of just some of the phases required to perform HIL testing:

- Control of the test sequence to run and ability to regressively test
- Injection of test stimulus in real-time
- Access to the model parameters and data and control of the simulation state
- Access to external devices (Testers, Voltmeters, etc.)
- Access to and control of PCM interfaces (diagnostic tools)
- Ability to generate test documentation and save test data
- Test data analysis during the test

The system chosen to perform the testing must provide the power to perform the above tasks, yet needs to be relatively easy to use in order to minimize test development costs and allow for changes in the testing process. The system chosen by Visteon is the software tool ControlDesk™, by dSPACE. ControlDesk provides all of the capabilities needed for a HIL test system, including hardware/model management, data acquisition, and graphical user-interface capabilities. The Test Automation extension of ControlDesk is based on the python language, and it provides the interface libraries needed to automate the tools and programs needed to perform the above listed tasks.

![Figure 3 Interfaces in HIL System](image)

**Python is an open source language, available on the [www.python.org](http://www.python.org) website. It is an extremely powerful,**

[1] MATLAB, Simulink and Stateflow are trademarks of The MathWorks, Inc.

[2] ControlDesk is trademark of dSPACE GmbH.
object-oriented language with programming syntax and usage very similar to Basic. It has been said that python can speed up development by a factor of 10 over other programming languages, such as C [4]. Python provides all of the control constructs and modular programming functionality needed for the development and maintenance of complex test sequences. The ControlDesk software has an integrated python interpreter and a function selector tool, to make python development as easy as possible for the user. Test scripts can be run directly, or they can also be connected to events, including GUI events such as Command buttons. Visteon has been able to program very complex test sequences and make them functional in very short periods of time.

Test stimulus generation

ControlDesk also provides mechanisms to assist in the generation of test stimuli (data streams) for specific test runs. The test stimulus is generated through python, using a library designed to generate the stimuli. In order to generate this data in real-time, software and hardware mechanisms have been built into ControlDesk and the dSPACE hardware, which allows for full fidelity of stimulus generation, with time based events guaranteed to occur during testing. A sequence generator tool is included with ControlDesk, which allows for interactive graphical user design of test stimuli. The sequence generator is based on time tags and specific function types, which can be combined in different manners. There are also control constructs, which allow for easy looping and conditional execution of the data streams. The tool also assists the user by providing a pre-generation of the data, for verification prior to the test, and then it provides the functions to allow the user to map the data to model variables and execute the stimulus both interactively and within test sequences. Another feature of the Sequence generator is the capability to playback recorded data streams, captured during either simulations or actual test runs. This capability has been used very effectively by Visteon to simulate drive cycles and control specific test sequences. Any data waveform stored in a pre-designed MAT file format can be used for this purpose. The dSPACE hardware and software systems provide the automatic buffering and interpolation processes that allow for real-time data playback during simulation runs. Using test stimulus to control test execution is an example of a data-driven system. This has many advantages over a model driven system, in that changes in the data streams do not require model changes and provide greater flexibility in the modeling process. The models are simpler to design with less real-time overhead, and this also allows for using the same models for both offline simulation and real-time testing.

Advanced test scripting capabilities

ControlDesk Test Automation provides libraries of functions for controlling the user interface and ControlDesk environment. It also has a Macro Recorder, which gives the user an easy method for capturing testing sequences performed through the GUI. The Macro Recorder saves these captured sequences as python scripts, which can then be easily replayed and also modified to extend and augment their functionality. There are interface libraries for communication protocols, such as RS-232 and GPIB, which allow for interfacing to external devices and systems. Libraries are also provided to allow for report generation and data capture into several windows COM OLE programs, such as Word and Excel. Finally, sets of functions are included which allow the user to access and control diagnostic devices and calibration tools, using either standard protocols or the ASAP/MCD standards [3].

Future path

Python is an extremely powerful tool and can be used to do every one of the tasks required for test automation. Visteon has had a successful experience with the use of test automation because of this. The next step in the future involves using a new 4GL testing system from dSPACE, called AutomationDesk, which is based on graphically designing test sequences. The system is still based on python, but it eases the development and maintenance process by providing a more intuitive graphical design environment, as opposed to writing code scripts. It is not a code generator, but instead generates testing process objects from clearly defined test system classes. AutomationDesk also provides a very powerful test management functionality, which allows for more robust management of all of the facets of a test. These include the scripts, models, test data, test results, stimulus and parameter data, etc. [5].

FAILURE INSERTION AND CLOSING THE LOOP

General overview

In order to fully test a PCM, the PCM must be connected to hardware interfaces that simulate the real plant environment as closely as possible. The PCM is generally designed to test these interfaces for any failures, such as short circuits, missing sensors, etc, at which time the PCM will store a diagnostic code in its memory. A large percentage of the actual code in a production PCM is for diagnostics. If one of these failures occurs, the PCM software is designed to recover from this failure and ensure the safety of the passengers and the environment. The dSPACE HIL system provides a configurable set of relays in a Failure Insertion Unit (FIU), which can be commanded to simulate these failures. A failure for a PCM input can generally be simulated in software and standard hardware. By sending a 0, 5 or vehicle battery voltage
input from simulated sensor different types of sensor failure can be simulated. To simulate failures for PCM outputs, however, this involves using electronic relays to provide open circuits, short circuits to battery voltage or ground, or even pin-to-pin short circuits. In addition, PCM outputs generally expect to have a specific load attached, and failures are triggered if this is not the case.

1. Place PCM in desired performance state
2. Inject a failure
3. Evaluate the PCM performance during the fault condition
4. Debrief the diagnostics tool to determine if the fault was properly detected
5. Reset the fault codes to allow for further testing

The above test sequence is fully repeatable and by using it, full lights-out testing sequences are possible. All of the diagnostic codes for a PCM may be evaluated in a single test run (with a combination of sequences) and a complete test report will be automatically generated to record the results. Once this process is implemented, the daunting task of test regression for iterative software releases is easily contained.

**BENEFITS – HIL IN EVERY DEVELOPMENTAL PHASE**

With our experience during the developmental and implementation stages of the HIL systems we found several advantages of using HIL systems in testing and validating our software product. Although, the initial scope for using the HIL system was limited, we now use the system in every phase of the PCM software development process. Figure 5 illustrates the System Engineering V with the phases of software development and highlights how HIL is being used in every stage of the software development.

Broadly, the HIL applications for PCM testing in the design phase can be classified as Open Loop Testing (OLT) and Closed-Loop Testing (CLT). In the OLT a
simpler I/O model is used with no feedback from the plant model, as shown in Figure 6. In the OLT the inputs are not dependant of the outputs of the system. They are independent of each other, and consequentially the dynamic behavior of the system cannot be tested in OLT. In OLT a set of values for inputs are provided to the system and the output responses of the systems are monitored and recorded.

In OLT the performance of the system software, actuators commands or outputs from the PCM are monitored and verified, while feeding a predefined set of values of sensor data to the PCM. OLT is applicable to check the PCM I/O, low-level drivers, unit testing and some integration testing. On the other hand, the CLT requires a complete plant model with appropriate feedback response for the system as shown in Figure 7. In a CLT environment the PCM commanded values are used to run the actuator plant models. The actuator plant model output is in mechanical or electrical units and is further used in the plant model to compute the plant behavior and dynamic sensor responses.

In the CLT environment the controller feedback loop is preserved, and thus a complete system verification and validation is possible in this environment. A simple driver model is used to drive the vehicle simulation where vehicle speed is monitored and the throttle and brake command are changed manually or using an automatic driver controller model. Closed loop simulation is well suited for system level testing and verification. In the following sections these applications of HIL are discussed in detail.

EXECUTABLE I/O TESTING USING OLT

Implementation

Development of Low-Level Input/Output Driver (LLD) software is one of the early phases of PCM software development. The LLD is implemented on top of the Real Time Operating System (RTOS). The LLD software is the interface between the PCM Hardware and/or High Level Driver Software and/or the application software. The LLD software is a fundamental piece of software in order for the PCM application software to work properly with the system hardware.

Traditionally, testing of the LLD software is done on what is referred to as a static test bench. The input to the hardware is provided from a power supply and/or a signal generator and the outputs of the hardware are monitored with a chart recorder and/or an oscilloscope. An early phase of automated testing of LLD software was implemented by a skeleton executable software layer developed on top of the LLD software layer; hence it is called the “Executable I/O (EIO) software”.

The different layers of the EIO software and the HIL system interface block diagram are illustrated in Figure 8. The executable layer is implemented in two level structures, the first is the communication layer and other is the command layer. The communication layer interfaces with the user and the command layer exercises the LLD software to manipulate the hardware I/O. The communication layer of the EIO software is integrated with the HIL communication layer, and the HIL becomes the server and the EIO software becomes the client. A sequence of tests is developed to exercise the complete range of all I/O. The HIL system Input drivers command the EIO software to provide a complete range of valid as well as invalid values to output drivers. The HIL system co-ordinates the data playback and data capture simultaneously to completely automate testing of LLD software. This EIO software implementation provided the capability of testing the LLD software with test points over a range of the inputs and outputs.

Prior to the integration of HIL with the EIO testing, the manual mode of testing was only capable of using a very limited set of data points, which was not exhausting enough to completely test the entire range of the hardware I/O. This manual method of testing was shown to be both time consuming and error prone. Using HIL
technology for the EIO software testing the complete I/O range can be tested automatically.

This implementation of HIL integrated EIO software significantly reduced the testing time of all PCM I/Os, and provided a far more complete range of every test as well. In later applications, using this testing very early in the development phase flagged some discrepancies between the requirements, the specifications and the software implementation. By discovering these discrepancies at this early phase, significant resources were saved, compared to finding these discrepancies later in the development cycle.

UNIT LEVEL TESTING USING OLT

The next level of testing phase is Unit Level Testing (ULT) of application software. ULT is a derivative of OLT. The Powertrain application software consists of multiple interdependent features, and each feature contains different level of sub-features. The features and sub-features of application software are defined by functional boundaries. The features and sub-features are developed based on the Powertrain requirements. The features and sub-features are tested against such defined requirements. In ULT one piece or one unit of software defined by certain functional boundaries is tested. For ULT, sets of inputs are designed to exercise the specific unit of software and fed into the PCM, and the output from that specific piece of software is monitored and recorded. The outputs of the software are checked against the requirements for the software given that set of inputs. Traditionally, the ULT is a manual operation using tools like a software debugger and manually stepping through software code to check the values of the outputs of the software, as well using tools like an oscilloscopes and chart recorders to record outputs from the PCM. The HIL integration of the ULT provided a more methodical way of completing unit test on a target. Utilization of test scripting and test stimulus generation technique the HIL system provided some level of dynamic input to the system, rather than just a static set of input values.

INTEGRATION TESTING USING OLT

Once, sets of ULTs are completed, the next obvious step is to recombine these units and test on a sub integration level. The integration of features is tested using the OLT methodology to test the interaction between several features. In the sub-integration level the complete software is not integrated with all the required features, but a few closely related features are integrated as a broad feature. In this test environment the feature level inputs are provided from the HIL system as dynamic inputs, and the outputs from the sub-integration level are monitored and recorded. Even though the input values may have some dynamic behavior in it, the absence of dynamic plant model prevented it from running in closed loop environment. The dynamic inputs were not based on the system response output; rather it was a coherent set of input data. Refer to Figure 9 for a block diagram of open loop integration testing. The next natural progression of testing is the verification of the complete feature in Closed Loop.

PROGRESSION FROM OPEN LOOP VERIFICATION TO CLOSED LOOP VERIFICATION

In the OLT methodology of verification, the inputs to the PCM are provided during the simulation, and the actuator outputs from the PCM are monitored. The control loop feedback mechanism is not in place so the dynamic response of the PCM software cannot be tested in OLT. CLT testing is necessary in order to fully validate the functionally of the PCM.
SYSTEM LEVEL TESTING USING CLT

The components of system level testing using CLT are illustrated in Figure 10. The fundamental difference between the OLT and CLT is the presence of dynamic plant model in the CLT system running in real time simulation. The plant models represent the behavior of the actual plant and provide appropriate feedback response. The response of the plant model influences the input to the system, thus providing a complete dynamic system simulation with controller feedback loop preserved. Figure 11 shows a block diagram representation of Open and Closed loop testing interfaces of the HIL system with the PCM. The “Ambient_and_Environment” block contains static inputs such as atmospheric pressure and temperature; the “Driver_Inputs” block contains a manual and automatic driver model. The automatic driver model compares the vehicle speed feedback with the speed stimulus and decides the required throttle and brake inputs to the plant model, while manual driver provides the hooks to control the brake and throttle pedal input from external source under simulation environment.

The “PCM_Interface” block contains the routing of I/O signals between a specific family of PCM and the HIL system hardware. The "Vehicle Model" contains the actual plant model representing the dynamic behavior of the plant, for our application it is a combination of and engine and transmission model. We have implemented a complete set of sensors, actuators and plant models in this system. In order to be able to simulate in real time, we used a regression based engine model, using mapping data for the same engine from a Dynamometer. Simplified transmission, driveline and vehicle models were also implemented. After successful simulation of Key On Engine Off (KOEO) and Key On Engine Running (KOER) conditions, the system was successfully tested and simulated for idle conditions, followed by manual and automatic drive cycle simulation implementation.

Prior to integration of the plant models with the HIL systems, the plant models were simulated and validated on the desktop. After validation of the plant model in the desktop environment the models were integrated with the HIL system. In order to validate the plant models the results obtained from a real engine drive cycle test conducted on a Dynamometer were compared with the simulation results. In drive cycle exercise the vehicle is driven through a known speed profile (e.g. FTP UDDS). The target speed is input to the automatic driver model. The automatic driver receives the actual speed of the vehicle as a feedback from the complete vehicle plant model. The automatic driver manipulates the brake and throttle to keep the vehicle in targeted speed. The vehicle speed profile from the Dynamometer test cell recording (using calibration tool, ATI Vision 1.9) was used as the input stimulus for the automatic driver for the simulation. Engine torque, engine speed and vehicle speed from the simulation are recorded using ControlDesk, the HIL user interface software, and the data was compared. The comparison (not presented in this paper) of the vehicle and engine speed profile and engine torque values along with other variables showed a very good correlation with minimal error. The maximum errors in these comparisons were less than 10 percent. Although final software verification and validation can be done only on a real vehicle, the complete closed loop simulation can easily be used for validating software at a system level during the early design stages of software development with less than 10 percent error margin to validate the PCM software.

There are certain limitations in using HIL for validation and verification of PCM software due to high
computational requirements for a real-time simulation. For example the emission related testing and validation is not feasible in the current HIL environment. A better plant model (e.g. complete detail combustion based model), however, can provide better correlation of engine variables as well engine performance. However, a detailed combustion based model may not be suitable for real time execution with the present computing technology.

Future advancement, both in real-time computing and plant modeling will close this gap of complete bench top simulation and may make the HIL an absolute replacement for a real vehicle for software validation and verification.

CONCLUSION

A brief discussion of on the implementation and usage scenario of HIL systems at Visteon Corporation for PCM software verification and validation is presented in this paper. The Powertrain Software development group in Visteon uses HIL systems in every step of the software developmental phase.

Using HIL in integrated EIO software, Unit Testing, and closed loop testing of the system software significantly reduced the testing time and improved software quality. Software quality is improved by identifying potential errors and/or discrepancies between the requirement, specification and software very early in the development phase.

The successful implementation of closed loop simulation has proved to be useful in validating several features at the system level. However, further improvements in real time computing technology and implementation will be required for simulating features such as emission requirements.

REFERENCES


3. S. Nabi, “Development of an Integrated Test Environment for ECU Software Validation,” dSPACE User Conference 2002,


5. Lamberg, K.; Richert, J.; Rasche, R.; A New Environment for Integrated Development and Management of ECU Tests, SAE 2003-01-1024, Detroit, USA.

6. Gehring, J.; Schuette, H.; A Hardware-in-the-Loop Test Bench for the Validation of Complex ECU Networks, SAE 2002-01-0801, Detroit, USA

CONTACT

Syed Nabi
Visteon Corporation
Visteon Technical Center
17000 Rotunda Dr.
Dearborn, MI 48121
Phone: (313) 755-1881
E-mail: snabi@visteon.com

Mahesh Balike, PhD
Visteon Corporation
Visteon Technical Center
17000 Rotunda Dr.
Dearborn, MI 48121
Phone: (313) 755-3804
E-mail: mbalike@visteon.com

Jace Allen
Senior Project Engineer
Applications Group
dSPACE, Inc.
28700 Cabot Drive
Suite 1100
Novi, MI 48377
Phone: (248) 567-1265
Fax: (248) 567-0130
E-mail: jallen@dspaceinc.com

Kevin Rzemien
Supervisor
Applications Group
dSPACE, Inc.
28700 Cabot Drive
Suite 1100
Novi, MI 48377
Phone: (248) 567-1265
Fax: (248) 567-0130
E-mail: krzemien@dspaceinc.com

ADDITIONAL SOURCES


2. H. Hanselmann, “Hardware-in-the-Loop Simulation Testing and its Integration into a CACSD Toolset”,


DEFINITIONS, ACRONYMS, ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN</td>
<td>Controller Area Network</td>
</tr>
<tr>
<td>CLT</td>
<td>Closed Loop Testing</td>
</tr>
<tr>
<td>DUT</td>
<td>Device Under Test</td>
</tr>
<tr>
<td>ECU</td>
<td>Engine Control Unit</td>
</tr>
<tr>
<td>EIO</td>
<td>Executable I/O Software</td>
</tr>
<tr>
<td>FIU</td>
<td>Fault Insertion Unit</td>
</tr>
<tr>
<td>FTP</td>
<td>Federal Test Procedure</td>
</tr>
<tr>
<td>HIL</td>
<td>Hardware-In-The-Loop</td>
</tr>
<tr>
<td>HW</td>
<td>Hardware</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>LLD</td>
<td>Low Level Driver Software</td>
</tr>
<tr>
<td>MBD</td>
<td>Model Based Development</td>
</tr>
<tr>
<td>OBD</td>
<td>On Board Diagnostics</td>
</tr>
<tr>
<td>OEM</td>
<td>Original Equipment Manufacturer</td>
</tr>
<tr>
<td>OLT</td>
<td>Open Loop Testing</td>
</tr>
<tr>
<td>PCM</td>
<td>Powertrain Control Module</td>
</tr>
<tr>
<td>RTOS</td>
<td>Real Time Operating System</td>
</tr>
<tr>
<td>UDDS</td>
<td>Urban Dynamometer Driving Schedule</td>
</tr>
<tr>
<td>ULT</td>
<td>Unit Level Testing</td>
</tr>
</tbody>
</table>