Automatic Simulation Measurement and Verification of Inputs and Outputs of Body Control Module

Vinaya CH
School of Electrical Engineering
Vellore Institute of Technology, Vellore
Chennai, India

Bagyaveereswaran .V
Professor, Select
Vellore Institute of Technology, Vellore
Chennai, India

Abstract:- An Electronic Control Unit is an embedded system that controls one or more electrical system in a motor vehicle. Body control module (BCM) is one of most significant unit responsible for monitoring and controlling main electronic accessories (exterior lights, interior lights, door locking, immobilizer etc) related to body of a vehicle. Manually testing the system is time consuming process. In this project a new method is proposed to automate the testing process using DSPACE and Python .

***Keywords**:- Body control Module, Flex ray, DSPACE, MATLAB/SIMULINK, Control Desk, Python.*

I. INTRODUCTION

In automotive electronics, electronic control unit (ECU) is a generic term for any embedded system that controls one or more of the electrical system or subsystems in a motor vehicle. The electronic control unit (ECU) acts like the brain of the automobile: it receives signals from the various sensors and decides if and when each actuator should be deployed. These ECUs are typically located in the middle of the vehicle or beneath the front seat, where it is well protected. Types of ECU include Electronic/engine Control Module (ECM), Electric Power Steering (EPS),Instrument Panel Cluster(IPS), Brake Control Module (BCM or EBCM), Central Control Module (CCM), Central Timing Module (CTM), General Electronic Module (GEM), Body Control Module (BCM), Steering Switches Module (SSM), Radio Receiver Module(RRM), Infotainment Module (ITM),control unit, or control module. Taken together, these systems are sometimes referred to as the car's computer. Sometimes one assembly incorporates several of the individual control modules (PCM is often both engine and transmission). Due to advanced technology, some vehicles which are equipped with advanced air bag systems, the ECU can also receive input from additional sensors that detect occupant weight, seating position, seat belt use, and seat position to determine the force with which frontal air bags should deploy. The performance of ECU depends on vehicle capability. They have ability to self-diagnosis and repair errors. For eg: If the ECU identifies a problem, the air bag readiness (warning) light will illuminate on the instrument panel. When the air bag readiness light is illuminated, the air bag system might not perform properly in a crash. A body control module may control one or more systems. The body control module which tested in this project controls 40 outputs. So manually testing the whole performance of a controller is time consuming. So automation is needed. In this project a new method is proposed to automate the testing process using DSPACE and python. First, we have to model the environment parameters using MATLAB/Simulink. Then, manual testing using CANoe, control desk and DSPACE. After that automate the testing process using python programming.

A. Tools And Software Used

- *CANoe*

CANoe is widely using for development and testing. It is from Vector Informatik GmbH. CANoe is mainly using in automotive field. CANoe is using for development, analysis, simulation, testing, diagnostics and start-up of ECU networks and individual ECUs. Its widespread use and large number of supported vehicle bus systems makes it especially well-suited for ECU development in conventional vehicles, as well as hybrid vehicles and electric vehicles. The simulation and testing facilities in CANoe are performed with CAPL, a very interactive scripting language. CANoe supports CAN, LIN, FlexRay, Ethernet and MOST bus systems as well as CAN-based protocols such as J1939, CANopen, ARINC 825, ISOBUS and many more. CANoe is a tool that supports the entire development process for Networked systems

- *DSPACE DS 1006 :*

The DS1006 Processor Board is based on a single-core or a multicore AMD Opteron TM processor. This real-time processor (RTP) is the main processing unit. It can access modular I/O boards via its PHS bus. It is multiprocessing-capable via the (optional) DS911 Gigalink Module. The DS1006 Processor Board provides the following features. Single-processor system means that your real-time application is running on one single-core DS1006 board or one processor core of a multicore DS1006 board. Multiprocessor systems means that your real-time application is running on two or more single-core DS1006 boards, on two or more processor cores of a multicore DS1006 board, on two or more multicore DS1006 boards, or a combination of all these. The DS1006 uses a single-core or multicore AMD OpteronTM processor as the real-time processor (RTP). The RTP calculates your real-time models and accesses the I/O boards via the PHS bus. The processor has a temperature sensor to prevent the processor from overheating. During over heating software mechanism terminates the real-time application and enters low power mode. At an even higher temperature, a processor-internal mechanism (called AMD thermtrip) shuts down the CPU clock. This processor has Hyper Transport TM technology connection for high bandwidth and low latency access to I/O devices. Built-in memory controller for high-speed memory access.

II. MODELLING

The first stage is modeling. The test environment is created using MATLAB simulation with DSPACE cards. The outputs related to body of the car are modeled using MATLAB simulation . There are 40 outputs related body control in this project. The outputs are divided into three types. Analog output, digital output and PWM output. By modeling, these outputs are assigned to different cards in DSPACE. DS4004 and DS2202 are the 2 cards used in this project. Each output has a port number and a channel number.

III. MANUAL TESTING

After modeling the outputs, we need to test the CEM manually. For that DSPACE, Control Desk 5.4, CANoe are the tools we used. Through flexray we can give the diagnostic request message using CANoe diagnostic console. Then check the output voltage using multimeter. If the expected output voltage and actual output voltage are same, then we can conclude that CEM is controlling properly.

IV. AUTOMATING THE TESTING PROCESS

After manual testing we need to automate the testing procedure. For that DSPACE, Control Desk 5.4, are the tools we used and Python is the software we used. Developed the python script for automation and run the script using Control Desk 5.4 in DSPACE HIL setup.

V. BLOCK DIAGRAM

The above figure shows the block diagram for working of Central Electronic Module. A supply voltage V_b is provided for CEM. When we give an instruction to turn on or off an output by pressing switch at driver side of a vehicle, that instruction will go to BCM. BCM is the controller .So after receiving the instruction from driver side, BCM will give proper control signal to the particular output to turn on or turn off. This is happening in a real vehicle.

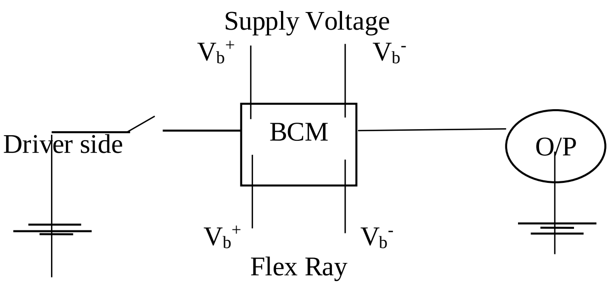


Fig 1:- General block diagram for diagnostics

The above figure shows the block diagram for working of Central Electronic Module. A supply voltage V_b is provided for CEM. When we give an instruction to turn on or off an output by pressing switch at driver side of a vehicle, that instruction will go to BCM. BCM is the controller .So after receiving the instruction from driver side, BCM will give proper control signal to the particular output to turn on or turn off. This is happening in a real vehicle.

The project is we need to test whether the BCM is giving proper control signal or not. This is an input output control testing. Instead of using the real vehicle, we are using DSPACE hardware in the loop (HIL) testing. During the testing, DSPACE will behave like a real time vehicle. In this project we are testing the body control module manually, and after that we are automating the testing process by using python script.

Since this is a testing, we are not giving instruction to BCM from driver side. Instead we are giving instruction through flex ray as diagnostics requests. Through diagnostics we can test whether the BCM is working properly or not. For diagnostics we are using flex ray. By using flex ray we are sending the diagnostic request and through flex ray we are reading the responses. If we are testing manually, we can use CANoe tool for sending diagnostics requests. If we are automating the testing process, by python scripting we can give the diagnostic requests. And the response and the results will print as an excel sheet.

Simulation done in the MATLAB using DSPACE cards are loaded in the DSPACE hardware. And by using control desk we can operate DSPACE. We have to give the whole conditions for testing environment using control desk tool. And we have to link with flex ray also. After that during manual testing, we can give the diagnostic request message using CANoe diagnostic console. To send diagnostic request, we are using 2F service. When we give a request message ,that message will go to BCM through flex ray. If the BCM receives the diagnostic request message successfully, BCM will send back a response message. That response message we can see through CANoe diagnostic console. After receiving the diagnostic request the BCM will give proper control signal to that particular output in DSPACE HIL setup. All inputs and outputs are modeled in HIL setup. So the control signal from BCM will go to that particular output block. After that we need to check whether the output is working according to the requirement. For that we can check the voltage across the output using multimeter. If the output voltage is equal to expected output voltage, the CEM is giving correct control signal and it is working properly. This is manual testing.

The body control module we are testing, controls 40 outputs. The outputs are divided into digital outputs and to make the CEM status into ‘control to application’. For a high side digital output, 1 indicate turn on and 0 indicate turn off. For PWM outputs, we have to test the CEM for different PWM’s. In this project we are testing for PWM’s 100%, 90%, 80%, 70%, 60%, and for 50% . That is ,through diagnostic request we will give instruction to controller to turn on the output in a particular PWM . Then we will check the output. If the output turn on with given PWM value, we can conclude that the central electronic module is giving proper control signal. So CEM is working properly for that output.

After manual testing, we need to automate the testing process. By automation, in a single step we can test all the outputs. Automation is done using python scripts. Diagnostic request for all the outputs can send through a single python script. The python script run through control desk. While running the script, the diagnostic request will send to central electronic module through flex ray. If the request reached successfully, the CEM will send back a positive response. And the CEM will send the required control signal to the corresponding outputs.

To compare with the expected output, we have to read the output voltage from flex ray. After reading we have to compare with the expected output voltage. If the expected output and the observed output is equal, result is pass, else result is fail. We have to print the output name, expected output, observed output and the result in separate columns in an excel sheet as a report. So during automatic testing, after running the script, result for all outputs will print in a single excel sheet. By automation work became simple, just we need to analyze the report. No need to measure the voltage of the outputs individually as in manual testing.

VI. MANUAL TESTING

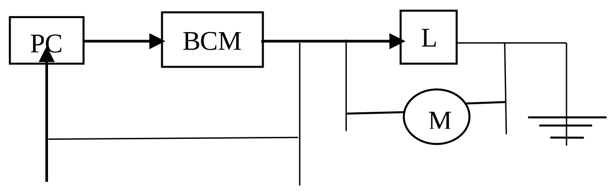


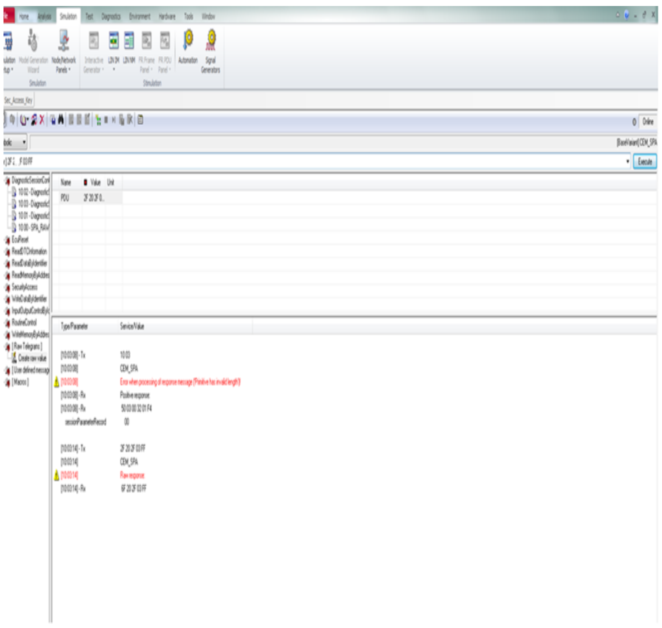
Fig 2:- Block diagram for manual testing

The above figure shows the block diagram for manual testing of Body Control Module(BCM). The project is we need to test whether the BCM is giving proper control signal or not. This is an input output control testing. During the testing, DSPACE will behave like a real time vehicle. During manual testing, we can give the diagnostic request message using CANoe diagnostic console. To send diagnostic request, we are using ox2F service. When we give a request message, that message will go to Body Control Module through flex ray. If the BCM receives the diagnostic request message successfully, Body Control Module will send back a response message. That response message we can see through CANoe diagnostic console. After receiving the diagnostic request message the BCM will give proper control signal to that particular output in DSAPCE HIL setup. All inputs and outputs are modeled in HIL setup. And each output is represented by an identifier. So the control signal from BCM will go to that particular output block. After that we need to check whether the output is working according to the requirement. For that we can check the voltage across the output using multimeter. If the output voltage is equal to expected output voltage, the Body Control Module is giving correct control signal and it is working properly. This is manual testing.

The Body Control module we are testing, controls 40 outputs. The outputs are divided into digital outputs and to make the CEM status into 'control to application'. For a high side digital output, 1 indicate turn on and 0 indicate turn off. For a low side digital output, 0 indicate turn on and 1 indicate turn off. For PWM outputs, we have to test the BCM for different PWM's. In this project we are testing for PWM values 100%, 90%, 80%, 70%, 60%, and for 50% . That is ,through diagnostic request we will give instruction to controller to turn on the output for a particular PWM . Then we will check the output. If the output turn on with given PWM value, we can conclude that the Body Control module is giving proper control signal for turning on. Then we gave instruction for turning off the output. Then check whether the output is turning off by using multimeter. After that we shift the controller working back to application. And check its output. If the output voltage is equal to expected output voltage we can conclude that BCM is working properly for that output. In this method we have to check controller performance for every outputs. This is manual testing and manual testing is time consuming. So we need to go for automation. That is automating the testing process

VII. AUTOMATED TESTING

The above figure shows the block diagram for automated testing for checking the working of Body Control Module. The project is we need to test whether the body control module is proper control signal or not. This is an input output control testing. Instead of using the real vehicle, we are using DSPACE hardware in the loop (HIL) testing. During the testing, DSPACE will behave like a real time vehicle. In this project we are testing the central electronic module manually, and after that we are automating the testing process by using python script.


```
Type/Parameter          Service/Value

[10:03:08] - Tx         10 03
[10:03:08]              CEM_SPA
[10:03:08]              Error when processing of response message (Primitive has invalid length!)
[10:03:08] - Rx         Positive response
[10:03:08] - Rx         50 03 00 32 01 F4
   sessionParameterRecord   00

[10:03:14] - Tx         2F 20 2F 03 FF
[10:03:14]              CEM_SPA
[10:03:14]              Raw response
[10:03:14] - Rx         6F 20 2F 03 FF
```

Fig 3:- Results of manual testing

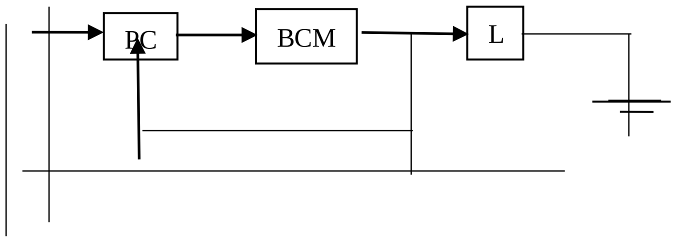


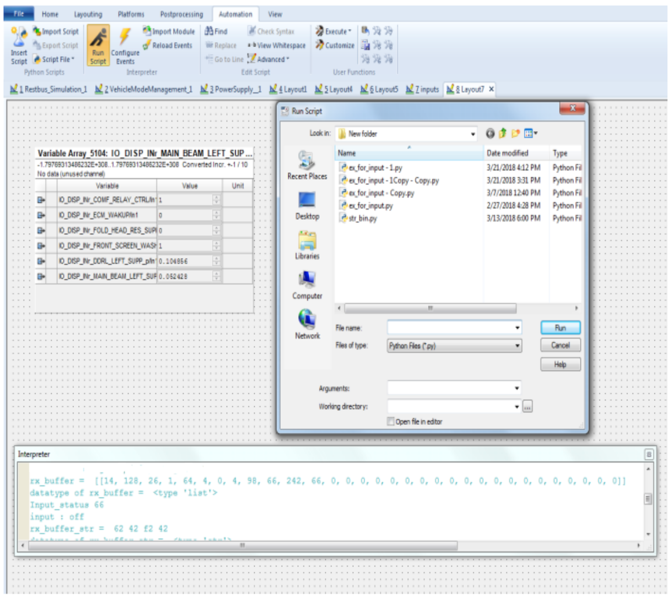
Fig 4:- Block diagram for automated testing

Simulation done in the MATLAB using DSPACE cards are loaded in the DSPACE hardware. And by using control desk we can operate DSPACE. We have to give the whole conditions for testing environment using control desk tool. And we have to link with flex ray also.

The central electronic module we are testing, controls 40 outputs. The outputs are divided into digital outputs and to make the CEM status into 'control to application'. For a high side digital output, 1 indicate turn on and 0 indicate turn off. For PWM outputs, we have to test the CEM for different PWM's. In this project we are testing for PWM's 100%, 90%, 80%, 70%, 60%, and for 50% . That is ,through diagnostic request we will give instruction to controller to turn on the output in a particular PWM . Then we will check the output. If the output turn on with given PWM value, we can conclude that the central electronic module is giving proper control signal. So BCM is working properly for that output.

After that, we need to automate the testing process. By automation, in a single step we can test all the outputs. Automation is done using python scripts. Diagnostic request for all the outputs can send through a single python script. The python script run through control desk. While running the script, the diagnostic request will send to central electronic module through flex ray. If the request reached successfully, the CEM will send back a positive response. And the CEM will send the required control signal to the corresponding outputs.

To compare with the expected output, we have to read the output voltage from flexray. After reading we have to compare with the expected output voltage. If the expected output and the observed output is equal, result is pass, else result is fail. We have to print the output name, expected output, observed output and the result in separate columns in an excel sheet as a report. So during automatic testing, after running the script, result for all outputs will print in a single excel sheet. By automation work became simple, just we need to analyze the report. No need to measure the voltage of the outputs individually as in manual testing.



Variable Array_5104: IO_DISP_INr_MAIN_BEAM_LEFT_SUP...

-1.79769313486232E+308..1.79769313486232E+308 Converted Incr. +-1 / 10

No data (unused channel)

Variable	Value	Unit
IO_DISP_INr_COMF_RELAY_CTRL/In	1	
IO_DISP_INr_ECM_WAKUPIn1	0	
IO_DISP_INr_FOLD_HEAD_RES_SUP	0	
IO_DISP_INr_FRONT_SCREEN_WASH	1	
IO_DISP_INr_DDRL_LEFT_SUPP_pIn	0.104856	
IO_DISP_INr_MAIN_BEAM_LEFT_SUF	0.052428	

Name	Date modified	Type
ex_for_input - 1.py	3/21/2018 4:12 PM	Python Fil
ex_for_input - 1Copy - Copy.py	3/21/2018 3:31 PM	Python Fil
ex_for_input - Copy.py	3/7/2018 12:40 PM	Python Fil
ex_for_input.py	2/27/2018 4:28 PM	Python Fil
str_bin.py	3/13/2018 6:00 PM	Python Fil

```
rx_buffer =  [[14, 128, 26, 1, 64, 4, 0, 4, 98, 66, 242, 66, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
datatype of rx_buffer =  <type 'list'>
Input_status 66
input : off
rx_buffer_str =  62 42 f2 42
```

Fig 5:- Results for automated testing

VIII. RESULTS AND OBSERVATIONS

In this project we test the controller manually and then automate the testing process. We got the correct results while testing manually and automatically. The output of manual testing is shown in fig.3. The flex ray diagnostic request and response message is shown in this figure. The output of automated testing is shown in fig.5. The output we can see in the interpreter of control desk and in an excel sheet.

REFERENCES

[1]. J. Novák and P. Kocourek, (2005) 'Automated Testing of Electronic Control Units Compatibility in Vehicle CAN Networks', IEEE ISIE 1423-1430.

[2]. Xinhong YANG, Yi LIN, Feng Gao , (2014) ' Automated Test System Design of Body Control Module ', IEEE 1542-1546.

[3]. Anna Bladh (2014), 'System Design of Automated Test Equipment for Electrical Control Units', Master Thesis Electrical Measurement Technology 20-72.

[4]. Bhagyashri U.Wani , Dr. Sunita P. Ugale (2016), 'Vehicle automation using controller Area Network', International Research Journal of Engineering and Technology 1195-1198.

[5]. Mr.Ajay Palkar, Ms.Priya Deshpande (2011)' Research Methodology for the Test Automation of Truck Controller Software System', International Conference on Environmental and Computer Science 104-111.

[6]. Fang Zhou, Shuqin Li, Xia Hou (2008) 'Development Method of Simulation and Test System for Vehicle Body CAN Bus Based on CANoe' IEEE Explore 7515-7519.

[7]. Xinhong Yang, Zhihua Yu, Mu Xiao, Guangbin Ji (2015) 'Automated Test System Design Based on Tellus for In-vehicle CAN Network', IEEE Explore (118–122).

[8]. www.vector.com/CANoe.

[9]. http://www.delphi.com/manufacturers/auto/body-and security/body/body_ _cntrl_modules.

[10]. Daehyun Kum, Joonwoo Son, Seonbong Lee,(2013),' Model-Based Automated Validation Techniques for Automotive Embedded Systems', IEEE Explore 2964–2968.