Research on Automotive UDS Diagnostic Protocol Stack Test System

Jinghua Yu and Feng Luo
Clean Energy Automotive Engineering Center, School of Automotive Studies, Tongji University, Shanghai, China
Email: thh1990@163.com, luo_feng@tongji.edu.cn

***Abstract*—This article presents the test methods of UDS (Unified Diagnostic Services) diagnostic protocol stack in vehicles, which includes two parts. One is the unit test of the main functions in network layer, another one is the functional test of the network and application layer. Corresponding test environments based on software and hardware are designed. Test cases are created according to international standard. All these tests ensure the efficiency of the development and the accuracy and standard of target protocol stack. Finally the designed test system is verified correctly.**

***Index Terms*—UDS, diagnostic protocol stack, unit test, functional test**

I. Introduction

Diagnostic system in vehicles is a monitor system, which supervises the running statuses of all the important components and reports them to users to ensure the security of a vehicle. Besides it is also the only interface between the ECUs in vehicles and the tools outsides. It means that this diagnostic system plays an important role in vehicles.

The UDS protocol stack is such a diagnostic system based on the international standards, which can implement vehicles' standard diagnostic communication normatively [1]. Nowadays more and more automobile companies and OEMs are using this UDS technique to unify their products and increase the efficiency of the cooperation with their customs or suppliers. Therefore the test for a specific UDS protocol stack is very important. It can ensure the correctness of the UDS stack according to corresponding specifications and international standards.

A good method can improve the efficiency of the process and find potential errors in a target stack in order to ensure that the UDS stack works correctly in vehicles.

Some methods have been designed to test UDS stack. A method using VECTOR's testing tools chain can test the whole protocol stack automatically [2]-[4]. Other software tools like LabVIEW and c++ studio can also be used to designed a simple upper tester of the test with some CAN simulation tools for a particular UDS stack [5]-[7].

Manuscript received July 16, 2015; revised September 13, 2015.

However, all these methods above can only be used after the whole stack is done. They cannot be implemented for a separated part of an UDS stack in a particular developing phase and not suitable for the stack under developing to check every step, which is essential to the development. Besides, the professional test tool for UDS is very expensive and the simple tester made by software like LabVIEW is not systematic. Therefore this article is going to introduce a method, which can make up for the defects above. It is systematic, flexible and cost-effective, which does not relay on a particle expensive tool.

II. Diagnostic Protocol Overview

The object of this test is the UDS Diagnostic Protocol Stack Based on CAN bus. According to corresponding international standards, the whole stack is divided into three layers in this article. Fig. 1 shows the structure of the diagnostic protocol stack.

Figure 1. Structure and UDS diagnostic protocol stack

The lowest layer is the driver layer of CAN communication based on ISO 11898. This layer is in charge of the transmission and reception of CAN frames [8].

The middle layer is network layer, which can pack the data from upper layer to the lower one, or unpack the data from the lower layer to the upper one [9]. There are two kinds of transmission in this layer. If the length of data is less than 9 bytes, it can be transmit via a single frame. Otherwise multi frames should be used. They obey the particular transmission mechanism specified in ISO 15765-2, including correct timing parameters, error handling and interface to the upper layer.

The highest layer is the diagnostic application layer. There are many services defined in ISO 14229-1 here, which can be called by requests. After successfully

doi: 10.18178/joace.4.5.388-392

receiving the data from network layer, the diagnostic layer can extract useful data and carry out relevant services. For example, a service called 'ECU Reset' can reset the target ECU when the operation condition is meet.

In order to test the diagnostic protocol stack introduced above, there are two kinds of test used in this article. They are unit test and functional test. The former one tests a single function in codes right after the codes are done, and the later one tests the real performances of the physical communication system.

III. System Design for Unit Test

Unit test is always the most basis part of a test in the software development and plays an important role. The purpose of it is to find the probable errors in every tiny software unit. It is an offline test, which is only based on a computer without any connection to the physical network. It can test every single function in codes.

A. System Structure

Fig. 2 shows the structure of unit test environment.

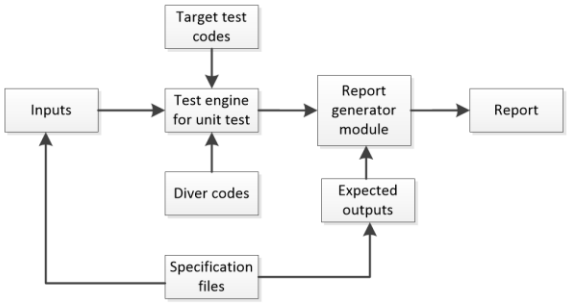

Figure 2. Structure of unit test environment

The test engine is the core of the unit test. It builds a virtual environment to run a part of stack and judge the correctness. The driver codes are extra codes to call the test codes. According to the stack specification the test inputs and expected outputs can be designed. After importing the designed inputs and expected outputs into the test system, a test report can be generated by the report module, which can compare the real outputs with the expected one and give the results.

This test can be used at any time to make sure that the codes written can perform as expected and lay a correct foundation to the following development of the stack.

B. Unit Test for Network Layer

The test engine of the unit test is set up by 'S-function builder' in MATLAB/SINMULINK. Users can establish the s-function based on c language easily through the graphic user interface offered by MATLAB. S-function is an abbreviation of system-function. It is actually an interface of MATLAB to get connect to the self-defined program written by users. By this way MATLAB can interact with other codes easily and flexibly.

The protocol stack codes can be called in MATLAB through 'S-function builder'. Fig. 3 shows the configuration dialog of s-function builder.

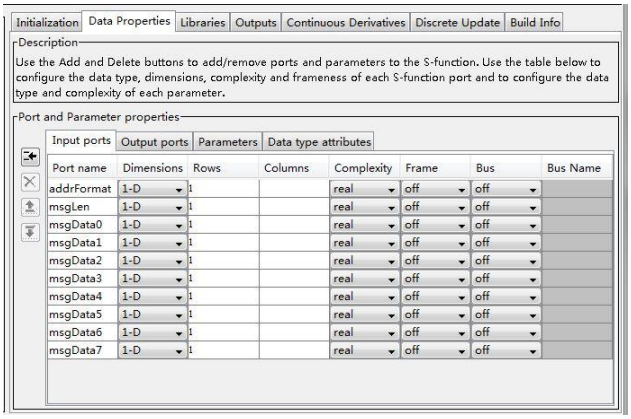

Figure 3. Configuration of s-function builder

Each function in the codes can be test as a test case. Before testing, inputs and expected outputs shall be designed according the expected function of the codes. After every execution of the test case, the real outputs will be compared with the expected outputs and then get the result whether the function is correct or not.

Fig. 4 shows the blocks of the test environment. The blue rectangles are the inputs and outputs interfaces connected with the report module. The biggest one in the center is the code engine, which does the main test work, together with some small data process module beside it.

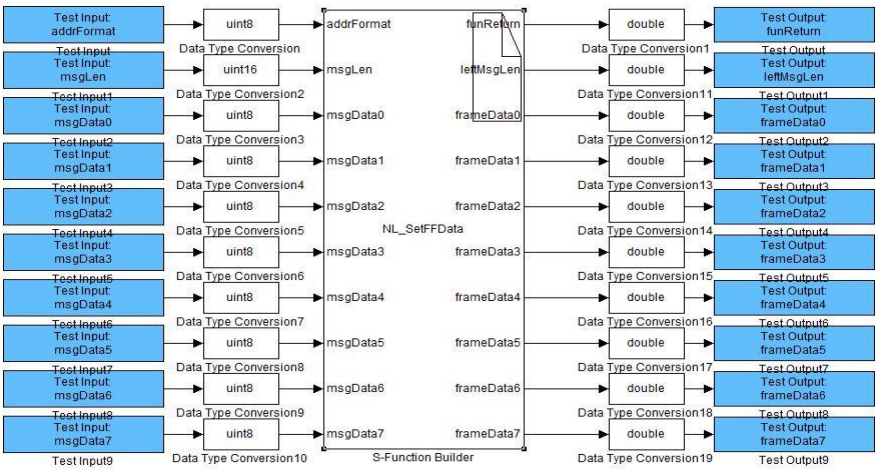

Figure 4. Example test environment for a unit test

The test case starts running when all the configurations are done in the report generation module. The system will get the inputs from the configuration file automatically and run the target test codes. Finally a test report is generated to indicate the end of this test case.

IV. System Design for Functional Test

The unit test can only make sure that each function has right outputs. But it cannot ensure the correct interaction between different functions. So it is necessary to do the functional test of the whole network layer. Whether the network layer works as defined can be checked through this test. It is an online test, which should be based on a UDS communication network. It can perform functional tests to judge if the stack works as descriptions in specification files.

A. System Structure

Fig. 5 shows the structure of functional test.

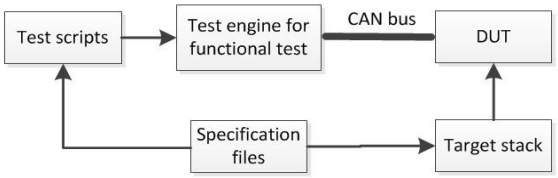

Figure 5. Structure of functional test

The test engine is a CAN simulation tool, which can simulate the communication according to the scripts. It drives the whole test and monitor the frames flow on the CAN bus. The target UDS stack is downloaded into an ECU, which becomes a DUT (Device under Test). The test engine and DUT constitute a real UDS communication network.

This functional test can be used to test every single function independently in network layer and application layer in UDS target stack at any time.

B. Functional Test for Network and Application Layer

The precondition of this functional test is a real network system. To build such a system, the MCU XDP512 from Freescale is used as the DUT and a computer with CAN simulation tool is used as the test engine. They are connected to the same CAN bus and work as a diagnostic system. Fig. 6 shows the real test environment of this functional test.

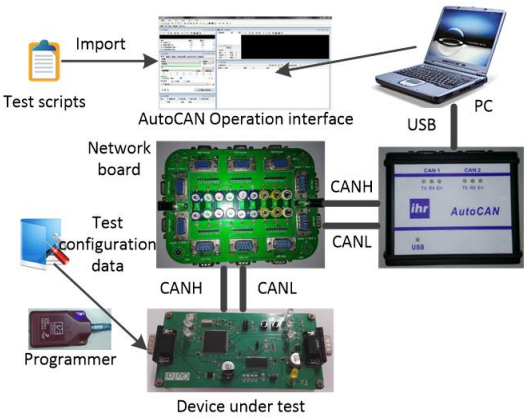

Figure 6. Test environment of functional test

The system includes test board of MCU, network board, which works as a CAN bus, relevant power, wires and programmer. The programmer downloads the configuration into the device under test to meet the specific test requirement. Compare to this part, the test scripts are imported into AutoCAN to execute to the simulation.

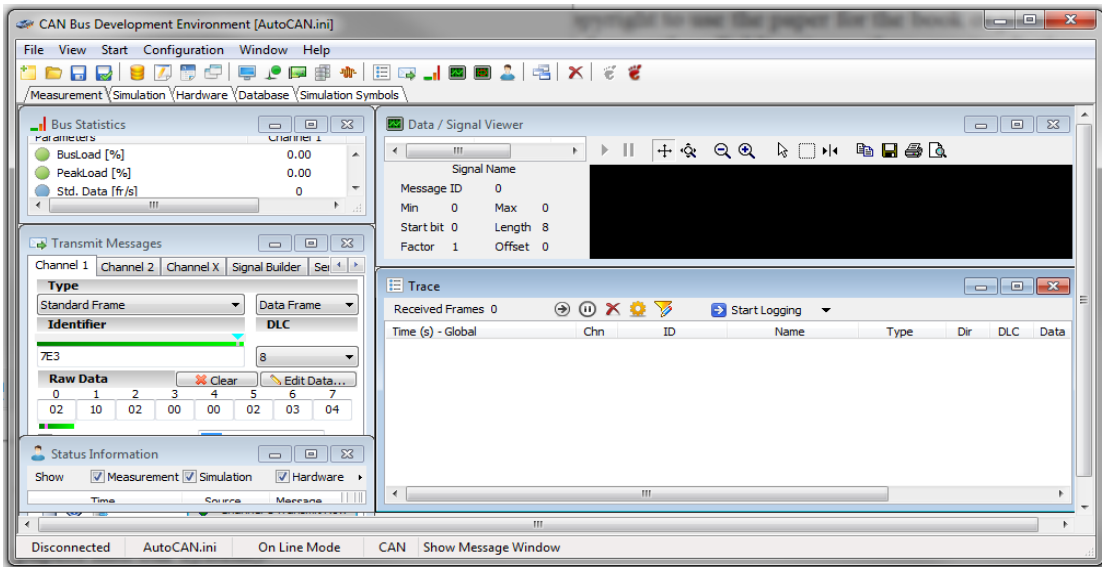

Figure 7. Operation Interface of AutoCAN

The connection between computer and CAN bus is made by a tool called AutoCAN. It is a tool for automotive CAN bus from ihr GmbH. It can be connected into the CAN bus directly and send or receive CAN frames through an up-level application program. Besides it also support the usage of scripts. By writing the scripts and simulating the codes, the computer can work as a simulation diagnosis instrument and meet the needs

of the network layer functional tests. Fig. 7 shows the operation interface of AutoCAN used in computer.

Besides the test environment, the test case also plays an important role in the test. A good design can traverse different situations and find potential errors to ensure the correctness of the codes.

The design of test cases in network layer is taken as an example. In order to traverse all the possible situations of the network layer, the parameters are divided into different layers.

Fig. 8 shows the structure of the parameter layer.

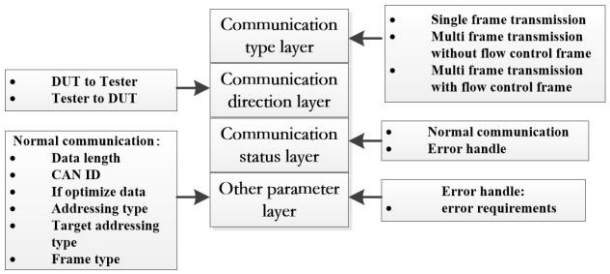

Figure 8. Structure of parameter layer

By changing a particular parameter and fixing the rest of them according to the layer structure above, a test case is built. A test case table can be generated by permutation and combination of these options, which can be used as the standard of the network layer functional test. Fig. 9 shows the format of the test cases table.

Single frame transmission				Multi frame transmission without control frame				Multi frame transmission with control frame			
UT->LT		LT-> UT		UT->LT		LT-> UT		UT->LT		LT-> UT	
OK	ERR	OK	ERR	OK	ERR	OK	ERR	OK	ERR	OK	ERR
The rest parameter list, which needs to be validated in different cases.											

Figure 9. Format of the test cases table

Each column in the table is a single test case, which tests a particular work situation of the stack under test.

Then the test scripts can be written based on the test cases using scripts editor of the CAN simulation tool. The scripts are based on C language. There are many functional APIs to control the behaviors of the tool. For example the function 'configure_message' is used to configure a CAN frame to a defined variable. Using 'send_frame' function can send this configured frame to the real physical CAN bus.

After establishing the test communication and running test scripts, the frames flow will be recorded by the tool. By analyzing the flow and test requirement, it is easy to judge whether the stack works correctly.

V. Test Verification

A. Verification for Unit Test

In order to verify the unit test system, a test case is introduced here as an example.

A function called 'NL_SetFFData' should judge whether the first frame in multi-frame communication is configured correctly. So the inputs are the data which needs to be transmit and the expected output is a correct data frame processed by this function.

Firstly a test data list should be designed. Fig. 10 shows the test case list of this function. The inputs are on the left side and the expected outputs are on the right side.

Inputs	#1	#2	#3	#4	Expected outputs	#1	#2	#3	# 4
addrFormat	2	3	4	2	funReturn	1	1	1	1
msgLen	9	9	9	5000	leftMsgLen	3	4	4	5000
msgData0	1	1	1	1	frameData0	16	20	3	0
msgData1	2	2	2	2	frameData1	9	16	16	0
msgData2	3	3	3	3	frameData2	1	9	9	0
msgData3	4	4	4	4	frameData3	2	1	1	0
msgData4	5	5	5	5	frameData4	3	2	2	0
msgData5	6	6	6	6	frameData5	4	3	3	0
msgData6	7	7	7	7	frameData6	5	4	4	0
msgData7	8	8	8	8	frameData7	6	5	5	0

Figure 10. Test case list of 'NL_SetFFData'

Then configuration file is filled according to this list. The test environment is shown in Fig. 4. After running the test, the report shows that the real outputs meet the expected outputs. Therefore the test for this function has passed.

By using this method, every function in the stack can be checked without delay and flexibly.

B. Verification for Functional Test

One test case from the list is also taken as an example to verify the functional test.

The test should include the following process: MCU sends a message with 30-byte-long data under the control of FlowControl frame. So the MCU should not only send a message of 30 bytes, but it should also cooperate with the computer according to the flow control mechanism. After importing the corresponding scripts and run the simulation, the UDS communication begins.

The expected frames flows should be as following. AutoCAN firstly receives a First-frame from MCU. After waiting specified time interval, it sends a FlowControl-frame to CAN bus. If MCU receives the FlowControl-frame correctly, it should send the consecutive frames until all the rest of data are sent. It should be focused whether the rest consecutive frames can be send continuously or they should wait further FlowControl-frames before sending.

The test tool records the frame flows during the test, which is shown in Fig. 11 and it shows the test process clearly. First the test tool received a First-frame from MCU at time stamp 452.54123. Then it sent the FlowControl-frame correctly at time stamp 525.54869. After MCU received this frame, it began sending its remaining frames until it finished. All the steps meet the requirement of the expected one. Therefore this test case has passed.

452.54123	1	0123	RX	8	10 1E 00 01 02 03 04 05
452.54869	1	0654	TX	8	30 08 14 FF FF FF FF FF
452.556222	1	0123	RX	8	21 06 07 08 09 0A 0B 0C
452.58122	1	0123	RX	8	22 0D 0E 0F 10 11 12 13
452.606213	1	0123	RX	8	23 14 15 16 17 18 19 1A
452.631221	1	0123	RX	8	24 1B 1C 1D FF FF FF FF

Figure 11. Records of CAN frame flows in this simulation

All the other test cases can be carried out by the method introduced before. It can check every single function of the UDS stack in both network layer and application layer.

VI. Summary

The test method introduced in this article is a useful and cost-effective one to test the UDS stack during the development. It designed based on international standard and independent from any particular test tool. Any CAN stimulation tool which supports scripts function can be used here. The unit test can check the codes on the computer without a connected network. And functional test can check the real performance of the stack. All these test methods can be performed at any phase flexibly to ensure that the development before is free of errors.

References

[1] Road Vehicles-Unified Diagnostic Services (UDS)-Part 1: Specification and requirements, International Standard ISO14229-1-2013.

[2] P. Philipp, T. Armin, P. Thomas, S. Mueller, and C. Raetz, "A quantitative study on automatic validation of the diagnostic services of electronic control units," presented at *IEEE International Conference on Emerging Technologies and Factory Automation in 2008*, 2008.

[3] S. Ma, Y. Xi, H. Ji, and F. Xu, "UDS Database development of ECU based on ODX," presented at the Annual Meeting of SAE-China, March 2013.

[4] H. Zhang, D. Zhan, and C. Lin, "Design of vehicle fault diagnosis system based on CAN bus," *Automotive Engineering*, vol. 30, no. 10, 2008.

[5] M. Salcianu and C. Fosalau, "A new CAN diagnostic fault simulator based on UDS protocol," in *Proc. International Conference and Exposition on Electrical and Power Engineering (EPE)*, 2012.

[6] X. Han and K. Bao, "Development and test of CAN bus network layer protocol stack," *Computer Engineering*, vol. 37, no. 15, pp. 232-234, 237, 2011.

[7] X. Chang, J. Yu, and Z. Liu, "Analysis and implementation of network layer of automotive fault diagnostic standard," *Automobile Technology*, vol. 9, pp. 40-44, 2006.

[8] Road Vehicles-Controller area network (CAN)—Part 1: Data link layer and physical signaling, International Standard ISO11898-1-2003.

[9] Road Vehicles-Diagnostics on Controller Area Networks (CAN)-Part 2: Network layer services, International Standard ISO15765-2-2004.

Jinghua Yu received B.E. degree in automobile engineering in 2013 from Tongji University, China. She is currently a Master student in School of Automotive Studies of Tongji University. Her current research is networks in vehicles, including CAN bus control and diagnostic in vehicles.

Feng Luo received his Ph.D. degree from Northwesten Polytechnical University. In 2002 he joined Tongji University, and currently is a professor in School of Automotive Studies of Tongji University. His research interests focus on automotive networks