Research and Implementation of the UDS Diagnostic System

Lingfeng Xie[1, a], Feng Luo[2, b]

[1,2]Clean Energy Automotive Engineering Center of Automotive Studies, Tongji University, China

[a]87xielingfeng@tongji.edu.cn, [b]luo_feng@tongji.edu.cn

Keywords: UDS protocol; CAN to Wi-Fi/Bluetooth; host computer; handheld terminal equipment.

Abstract. As an automotive diagnostic technology, the *UDS* (unified diagnostic services) protocol has been more and more widely applied. In this paper, a diagnosis system based on *UDS* is developed and tested. According to the *UDS* protocol, the software is designed and groups of diagnostic services are implemented. Besides, the host computer is designed so that the results of diagnosis can be parsed and displayed clearly. The host computer is divided into two types: one is implemented on *PC*s, which the communication mode is based on *CAN* bus; the other one is implemented on the handheld terminal equipment, the Android pad, which a *CAN* to *Wi-Fi/Bluetooth* hardware module needs to be designed so that the *CAN* frame can be translated from the microprocessor to the handheld terminal equipment. Finally, in order to verify the diagnostic system, we carry out some test cases and draw the conclusions that the diagnostic system is feasible.

Introduction

The *CAN* bus is the main standard in the current automobile network system of high speed. It supports the speed of 1M/s and distributed real time control with a high level of security and relatively low cost [1]. Nowadays, most of the vehicles use the *CAN* bus as their mode of communication.

The diagnostic technology in automotive refers to that acquiring the internal state information, finding the location of diagnosis and solving it in time without dismantling vehicles. With the widespread of the *CAN* bus, the *ISO* distributes a series of protocols to realize the standardization of diagnosis based on *CAN* bus. So the *UDS* (Unified diagnostic services) protocol was born [2]. With the help of this protocol, the efficiency of diagnose has been improved [3].

The *UDS* protocol is also called the *ISO* 14229-1 protocol [3], which has been established to define the common requirements and services for diagnostic systems in application layer. And the *ISO* 15765-2 [4] defines the transport protocol and network layer services. The *UDS* protocol allows that the diagnostic tester called client to build a communication with the diagnostic target *ECU* called server by *CAN* bus. The client mainly reads/writes the data from/to the flash of controllers, re-programming of *ECU*s, etc. The client translates the requests of diagnostic services defined in *UDS* and the sever carries out these diagnostic services. After carrying out these requests successfully, the server needs to return a possible message to the client, otherwise returning a negative message. The diagnosis services mainly conclude that reading *DTC* (Diagnostic Trouble Code), clearing *DTC*, reading and writing *DID* (DataByIdentifier) and 25 diagnostic services in total. Every service occupies their own *ID* named *SID* to differ from other services. And there are some sub-functions in most number of services to realize different functions.

System Architecture

In this paper, the diagnostic software based on *UDS* is developed [6,7]. At the same time, two host computers, which one is on *PC* and the other one is on handheld terminal equipment Android pad are also designed to test the diagnostic software. Via this diagnostic system and host computer, the diagnostic function of *ECU* can be implemented and tested.

The figure 1 refers to the diagnostic system with the host computer based on the *PC*. The system includes 4 parts: the *ECU* with diagnostic software, *CAN* bus, *SuperCAN* and host computer. The

SuperCAN helps the *PC* convert *CAN* frame data and serial data, so the communication between host computer and *ECU* can be realized via *CAN* bus.

The figure 2 refers to the diagnostic system with the host computer based on the hand terminal equipment. The system also includes 4 parts: the *ECU* with diagnostic software, *Wi-Fi/Bluetooth* model, *CAN* bus and host computer. The *Wi-Fi/Bluetooth* model helps the hand terminal equipment convert *CAN* frame data and *Wi-Fi/Bluetooth* data. The advantage of this system is the communication convenient and can be realized via wireless.

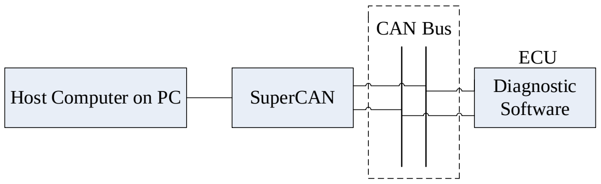


Fig 1.System architecture on PC host computer

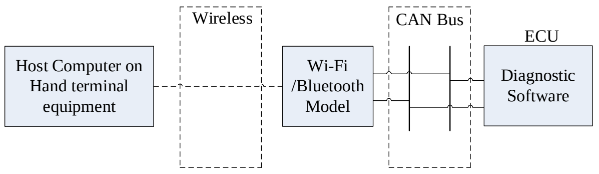


Fig 2.System architecture on hand terminal equipment

Diagnostic Software Design

Figure 3 refers to the software architecture. The software runs in a loop. Once a diagnostic request message is translated from the client, the server will catch the *CAN* frame and translate it to the network layer. The network layer will unpack and parse the *CAN* message, judging whether the message is single or multi frames, and whether the message needs to return a flow control frame. Then the rest messages will be translated to the application layer. The application layer of the server mainly unpacks and acquires the *SID* and sub-function *ID* to implement the specified diagnostic service. Finally, the application layer will pack and translate the state information to the network layer. The network layer will determinate to return single or multi *CAN* frames, then returning the message.

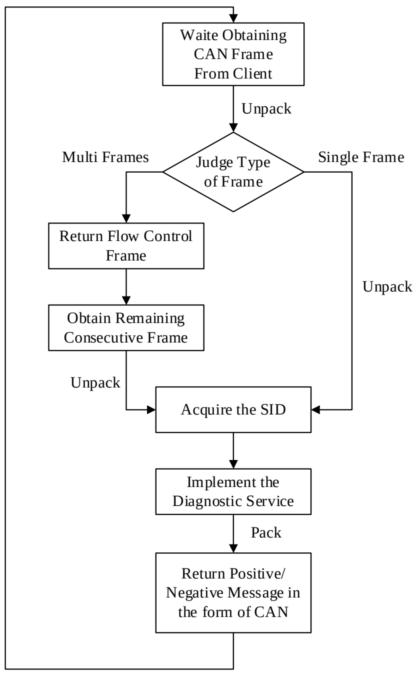

Waite Obtaining CAN Frame From Client
Unpack
Multi Frames
Judge Type of Frame
Single Frame
Return Flow Control Frame
Unpack
Obtain Remaining Consecutive Frame
Unpack
Acquire the SID
Implement the Diagnostic Service
Pack
Return Positive/ Negative Message in the form of CAN

Fig 3.Software architecture

Figure 4 shows the process of message transmitting. First and foremost, in the lowest layer, the physical layer, client and server are communicated via *CAN* bus. So the *CAN* model should be initialized, and some attributes are also set, such as baud rate. When receiving a *CAN* frame, the server will come into an interrupt and check the *ID* of the message whether is related to the diagnosis. The server will save the *CAN* message that is related to the diagnosis in its cache, and wait to be handled by the network layer. After implementing above operation, system exits the interrupt and waits for the next *CAN* frame.

In the network layer, an interface function should be defined so that the data can be translated between the network layer and application layer. The interface function will be called when the message is saved in the cache. Via the interface function, the software can realize the packing or unpacking function of the message. When translating message, the network layer determinates the message to translate in the form of the single or multi frames according to the length of the message in interface function. When receiving message, the network layer will unpack the *CAN* frame according to the protocol control information so that the application layer can understand which type of the *CAN* frame is belong to and which diagnostic services should be implemented.
The top layer, application layer, only handles the message from the network layer rather than need to consider the way of translation.

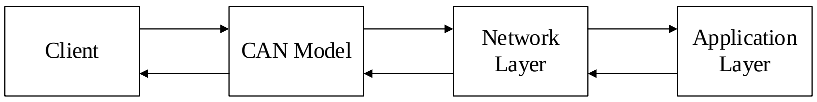


Fig 4. Detailed transmitting process

There are two forms of the transportation, which is judged by the network layer:

A. When receiving an unsegment message from client, the mode of transportation see figure 5.
B. When receiving a segment message from client, the mode of transportation see figure 6.

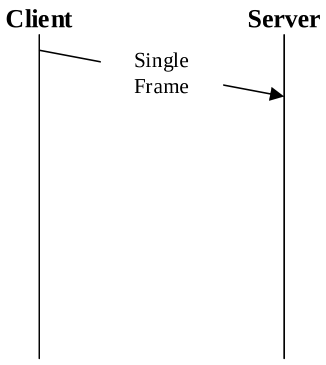


Fig 5.Receiving an unsegment message

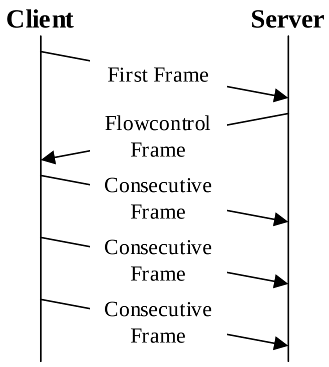


Fig 6.Receiving an segment message

Host Computer Design

After designing the software, the host computers are also developed. The host computer is designed according to the *UDS* protocol. Its function mainly concludes translating and parsing the diagnostic *CAN* frame return from the server to distinguish that which the diagnostic services are implemented and acquire some useful information. In this paper, two types of host computers are designed: one is based on the PC, the other one is based on the handheld terminate equipment.

A. The host computer based on *PC*

The host computer on the *PC* is developed on the basis of the *VisualStudio2012*. It can run normally on *Windows XP/7* operating system. The design ideas are divided in below 4 steps:
First, the *PC* is accessed to the *CAN* bus by using the *SuperCAN* because the *PC* can’t receive the *CAN* frame directly and the tool *SuperCAN* can translate the *CAN* frame into the serial data that *PC* can understand. Calling the library functions of the *SuperCAN* in this host computer so that the communication has been built.
Second, realizing the function of translation and reception of the *CAN* frame according to the *CAN* standard so that the client can filter the *CAN* frames from the serves and translate the diagnostic *CAN* frames to the target server via inputting the *ID*.
Third, realizing the diagnostic services defined in *UDS* protocol. The content and analysis of different diagnostic services are different. It is important to judge whether the frame is single or multi.
Finally, designing the operation interface, planning the layout of every component and realizing the trigger conditions of all diagnostic services to achieve a better human–computer interaction.

The operation interface are as figure 7. First step of operation is building communication, then inputting the *ID* of the server and choosing the specific diagnostic service, and fill in the relevant data information. After that, select the “*Execute Diagnose*” button, then the host computer will display the results of this diagnosis on the lower right.

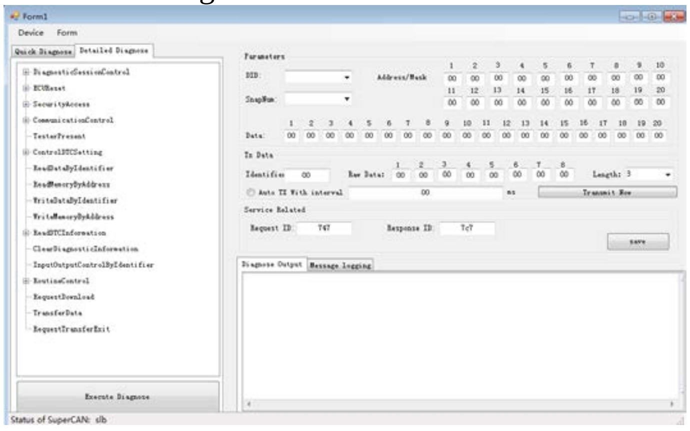


Fig 7.Operation interface on host computer based on PC

B. The host computer based on handheld terminate equipment

The host computer based on the handheld terminate equipment is developed on the basis of the android pad by Android Studio, because the Android *OS* is the most popular *OS* nowadays and the android pad is portable and easy to use. The operation interface on the android pad refers to figure 8.

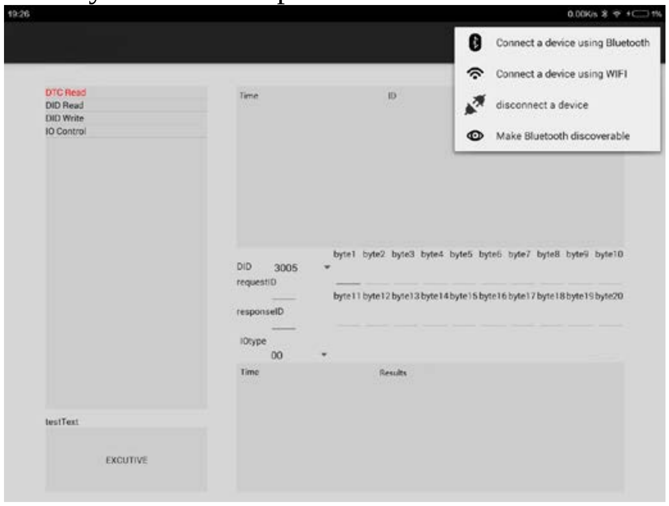
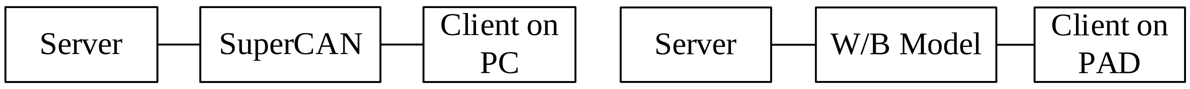

Connect a device using Bluetooth
Connect a device using WIFI
disconnect a device
Make Bluetooth discoverable
DTC Read
DID Read
DID Write
IO Control
Time
ID
DID
3005
requestID
responseID
IOtype
00
byte1 byte2 byte3 byte4 byte5 byte6 byte7 byte8 byte9 byte10
byte11 byte12 byte13 byte14 byte15 byte16 byte17 byte18 byte19 byte20
Time
Results
testText
EXCUTIVE

Fig 8. Operation interface on host computer on handheld terminal equipment

Because the android pad can’t receive the *CAN* frames directly and it supports the wireless communication *of Wi-Fi* and *Bluetooth*. Designing a hardware module that converting *CAN* to *Wi-Fi/Bluetooth* is a primary task. With the help of this module, the message can convert between the *CAN* data and *Wi-Fi/Bluetooth* data.
The architecture of this host computer is mainly divided into 3 layers: the bottom layer is used to configure the *Wi-Fi/Bluetooth* hardware modular and build connection so that the reception and translation of *CAN* frames can be achieved. The second layer is network layer that is used to pack and unpack the *CAN* frame. The top bottom is used to interactive with the users.
The workflow of the host computer based on android pad are basically the same as that on the *PC*s.

System Testing

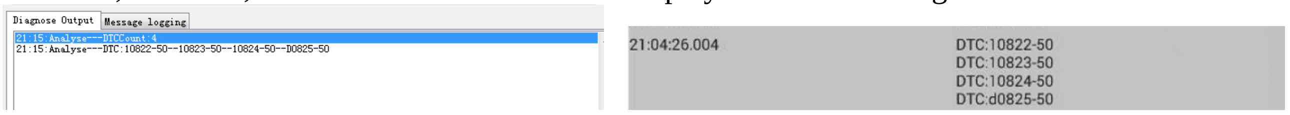
The testing systems shown in figure 9 is divided into two examples, one is based on the *PC*s and another one is based on the handheld terminal equipment.



(a).Testing system on PC (b). Testing system on handheld terminal equipment

Fig 9. Testing system

Setting one of the most common diagnostic service, *reportDTCByStatusMask* as an example, whose *SID* and the code of the sub-system are 0x19 and 0x02 respectively. Its function is used to read the current *DTC* in the server ECU. After building the testing system, considering that the situation of translating the multi *CAN* frames, making the server produce multi troubles, whose *DTC*s are 0x10822, 0x10823, 0x10824 and 0xD0825. The displays are shown as figure 10:



```
21:15:Analyse---DTCCount:4
21:15:Analyse---DTC:10822-50--10823-50--10824-50--D0825-50
```

```
21:04:26.004                DTC:10822-50
                            DTC:10823-50
                            DTC:10824-50
                            DTC:d0825-50
```

(a). Host computer on PC (b).Host computer on handheld terminal equipment

Fig 10. Display of the parsing return when multi trouble

From the figure 10, multi *DTC*s can also be displayed correctly both in the 2 host computers. After multiple groups of tests, the two host computers can display the right results.
Every diagnostic service defined in the *UDS* protocol has been tested with the two host computers. And all the services has passed the test.

Summary

As a standard for automobile diagnostic protocol, the *UDS* protocol based on the *CAN* bus can judge where and which diagnoses take place more efficiently with the improving number of the *ECU* in the vehicle. This paper develops the software on the basis of the *UDS* protocol with two types of host computers to verify its feasibility. Finally, the diagnostic software system passes all the tests and proves its correctness and consistency with the *UDS* protocol.

References

[1] BOSCH CAN Specification 2.0.

[2] Junkui Huang, Jinrui Nan, Zhi Chai, and Cheng Lin. Analysis of UDS Diagnostic Service Applied on Vehicle ECU [J]. Applied Mechanics and Materials (2013).

[3] Miller Jason, Thomas Tim, Waldeck, and Bill. Challenges and Benefits to Adopting a Worldwide Diagnostic Protocol Specification[C]. SAE World Congress (2004).

[4] ISO 14229-1:2013 Road vehicles – Unified Diagnostic Services (UDS) – Part1: Specification and Requirements.

[5] ISO 15765-2:2011 Road vehicles – Communication over Controller Area Network (DoCAN) – Part2: Transport Protocol and Network Layer Services.

[6] Wajape, Mahesh, Elamana, and Nithin. Study of ISO 14229-1 and ISO 15765-3 and Implementation in EMS ECU for EEPROM for UDS Application [C]. 2014 IEEE International Conference on Vehicular Electronics and Safety (ICVES) (2014).

[7] Song Yan, Wang Tianran, Xu Aidong, Wang Kai, and Yang Zhijia. CAN based Unified Customizable Diagnostic Measure Research and Realization [C]. UKACC International Conference on Control (2012).