Unified Diagnostic Services Protocol Implementation in an Engine Control Unit

Panuwat Assawinjaipetch[1], Michael Heeg[2], Daniel Gross[2], Stefan Kowalewski[3]

[1]Thai-German Graduate School of Engineering
King Mongkut's University of Technology North Bangkok
Bangkok, Thailand
panuwat.assawinjaipetch@gmail.com

[2]Aachen Technical Centre, Ricardo Deutschland GmbH
Aachen, Germany
{michael.heeg, daniel.gross}@ricardo.com

[3]Lehrstuhl Informatik-11, Embedded Software Laboratory
RWTH Aachen Universityline
Aachen, Germany
kowalewski@embedded.rwth-aachen.de

***Abstract*—Engine control unit (ECU) is one of the most critical components in the automotive field. In the development phase, technology involving the design of the ECU varies greatly across manufacturers. Therefore, without a diagnostic standard, it can result in difficulties to calibrate and maintain the ECUs because only the developers have expert knowledge of their ECUs. Implementation of the diagnostic system in an ECU differs from manufacturer to manufacturer. Despite such a wide diversity, all ECUs are required to comply with common diagnostic standards so that they show the same behavior and provide common interfaces.**

A diagnostic system must, therefore, contain a protocol for connecting the diagnostic tools that the developers, testers and repairers use for checking the ECU's diagnosis information. Each protocol might be suitable for only one diagnostic system and vehicle components and systems need a great amount of effort to implement just a protocol for one particular diagnostic system. Nevertheless, there are many types of diagnostic systems defined by ISO (International Organization for Standardization) and SAE (Society of Automotive Engineers) depending on the type of systems and specific diagnostics from the vehicle manufacturers. As a developer, the author believes that this could be a never-ending problem each time the developer has to implement a system for specific requirements. Moreover, under some circumstances, the development time may more than double. To avoid that, the developer needs to develop a system that supports most of the diagnostic standards defined that called Unified Diagnostic Services (UDS) protocol. UDS protocol was defined by either ISO standard or SAE standard in order to support different diagnostic standards.

Keywords— UDS protocol, Unified Diagnostic Services

I. INTRODUCTION

In the field of automotive, ECUs are used for controlling mechanical components and devices in the engine such as throttle flaps, fuel injection systems, sensors and actuators. Moreover, ECUs control several kinds of system and behavior of the vehicle such as the internal combustion process, the braking system, tire pressure motoring system and so on. However, the ECU does not only control the engine, but also can detect, correct and prevent errors engines can encounter under real circumstances, for example miscalculation of the fuel/air ratio. The ECU will attempt to detect errors and report them to the user for each request. Author calls such mechanism diagnostic services.

Diagnostic services are defined by several standards by ISO, SAE, OEMs (Original Equipment Manufacturers) or even by the vehicle manufacturer themselves. For example, OBDII (On-Board Diagnostic II), SAEJ-1939 (Society of Automotive Engineers Japan) [6] and EOBD (European On-Board Diagnostic) are the well-known standards used in automotive fields. The UDS protocol was developed in response to the variety of diagnostic services. The UDS protocol is a concept proposed by ISO-14229 [2] standard to cover a big number of diagnostic service types.

The UDS protocol provides a number of necessary functionalities for repairers, developers and testers so that they can, for example, read or write data in ECU memory, program the flash memory and create specific behavior for an ECU such as provide a response on a timer interrupt event. As a result, the UDS protocol is very capable but also comprehensive and therefore complex in terms of implementing all of the ISO standard's protocol functionalities. Therefore, UDS protocol became a widely used protocol across the automotive manufacturers. However, UDS is a complicated protocol and is proposed as only a concept. As a result, there are differences in minor details for manufacturers to implement UDS protocol in their products. The goal of this paper is to simplify a solution and creates a solid concept of the implementation of UDS protocol in an ECU.

The paper is structured as follows. Section II presents the system overview of ECU. Section III describes the implementation of an ECU using OSI (Open Systems Interconnection) model. Section IV presents the results of an implementation of an ECU then presents the conclusion and future works for the regarded system.

II. SYSTEM OVERVIEW

An Engine Control Unit is a component that controls an internal combustion engine. There are functionalities that are implemented such as a diagnostic service, Air-Fuel Ratio

(AFR) control, combustion control, torque management and so on. However, there are differences in the functionalities between each manufacturer. Because of these differences, a standard protocol needs to be implemented in order to controls their functionalities to work in the same direction.
The application software inside an ECU can be simplified into two parts as shown in Figure 1 which are a diagnostic system and a control system. The control system is created for controlling a combustion engine. It has a capability to generate Diagnostic Trouble Codes (DTCs) if there was detected a malfunction. These DTCs provide status information of an engine. The DTCs are used by repairers or technicians for maintaining and diagnosing an engine. However, the repairer and technician must not have the capability to access the control system itself. Therefore, the diagnostic system is created to limit the access rights. The diagnostic system must provide complete information of a DTC and some part of the control system for calibrating purposes. As said, the diagnostic system in an ECU differs from manufacturer to manufacturer. Moreover, each diagnostic system uses a different set of DTCs. Therefore, UDS protocol is created with the capability to support various kinds of DTC that are created by ISO standards, SAE standards and vehicle manufacturers.

A. *Unified Diagnostic Services protocol*

The Unified Diagnostic Services (UDS) Protocol is a communication protocol in the automotive field, developed based on the idea of Keyword Protocol (KWP2000 [3]) to fulfill common requirements for diagnostic systems on CAN buses. The commands in UDS are divided into six groups according to their functionalities described in the following sections.

1) Diagnostic and Communication Management Functional Unit: The main function of the UDS Protocol. It specifies the ECU functionality, manages the diagnostic system, defines the user accessibility, controls the transmission of data etc.

2) Data Transmission Functional Unit: It is the functional unit that has the capability of accessing current data. Data must be read or written immediately after each request.

3) Stored Data Transmission Functional Unit: This group of functional unit allows Diagnostic tools to access the server's memory. The diagnostic tools can clear or read diagnostic information defined by the server's database inside memory.

4) Input Output Control Functional Unit: This functional group is used for controlling actuators, sensors, devices and equipment attached to the server. All of those inputs and outputs must be defined as Data Identifiers by OEM, Vehicle Manufacturer, etc., following the ISO standard.

5) Remote Activation of Routines Functional Unit: This functional unit contains a service that influences an ECU's behavior. In order to use this service, the routine identifier must be defined first by the vehicle manufacturer, the ISO standard, system supplier, etc.

6) Upload Download Functional Unit: The last group of functional units is used for requesting the negotiation of data transfer between a client and a server. This functional group contains services for requesting to start the transmission to manipulate data in an ECU's memory, for transferring data and for requesting to terminate a data transmission.

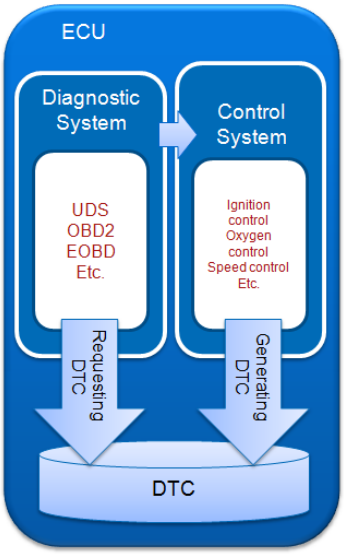

Figure 1: System Overview

B. *Diagnostics on Controller Area Networks (CAN)*

UDS protocol is only an application layer in the OSI model. Normally, the intercommunication inside various vehicles uses Controller Area Network (CAN) buses as a Physical Layer of the protocol. There are a number of standards used for different kinds of diagnostic services as mentioned in the Diagnostic Service section. For example, ISO-15765 is one of the principal ideas for developing a communication protocol of enhanced diagnostics on CAN buses. The standard defines a specification of the Network Layer and Transport Layer for Diagnostic Services. Moreover, this standard represents the latest protocol and is a mandatory standard for modern vehicles.

III. IMPLEMENTATION

In this section, we will provide an implementing concept of UDS protocol in an ECU that has protocols implemented while not overlapping with the others during a development. According to the current standard, UDS is a protocol that must be specifically developing to use with CAN buses. A CAN frame contains a data field and a CAN ID field. The values of these fields are determined by higher layer protocol such as UDS, Universal Calibration Protocol (XCP), CAN Calibration Protocol (CCP) and On-Board Diagnostics II (OBD2), which have a variety of frame formats. The OSI layer must be classified first for an efficient implementation. The classification of each protocol layer depends upon the frame format. Therefore, the OSI model needs to be defined in order to implement UDS protocol in an ECU. A simple OSI model that supports every protocol on the same CAN bus as shown in Table 1.

Application Layer	XCP	CCP	UDS	OBD2
Presentation Layer	N/A (Not Available)		N/A	
Session Layer	N/A		N/A	
Transport Layer	N/A		Diagnostic on CAN – Transport Layer (ISOTP)	
Network Layer	User defined		Diagnostic on CAN – Network Layer (ISOTP)	
Data Link Layer	CAN			
Physical Layer	CAN bus			

Table 1: OSI model for XCP, CCP, UDS and OBD2 protocol [1] [4] [5] [7]

A. Network Layer Development

The network layer must contain the ability to distinguish the type of protocols. XCP and CCP protocols are calibration protocols. Unlike diagnostic standards, a calibration protocol does not have a specific requirement for CAN ID definition. Therefore, the author of this paper advices that developers should define the address of vehicle components by using only 11-bit CAN identifiers for all calibration protocols. On the other hand, developers must define the address of diagnostic service protocols following the ISO standard and SAE standard, which uses 29-bit CAN identifiers, to give a full definition of packets. This method provides less concurrency of addressing between the calibration protocol and the diagnostic protocol. Packets of CAN frame should be stored in CAN buffers and remapped to a structure type variable for reasons of efficiency. Figure 2, illustrates the flow chart of the network layer that was developed by the author.

B. Transport Layer Development

The transport layer of the UDS protocol on the CAN bus is completely defined by ISO-15765. The transport layer uses the Protocol Control Information (PCI) byte to control the message flow. The PCI byte is a control byte that contains information on the frame type and frame type parameters. There are four types of frames, namely, Single Frame, First Frame, Consecutive Frame and Flow Control Frame. Response data from an ECU smaller than eight bytes, including the PCI byte, must use the Single Frame type to transmit messages. If response data exceeds eight bytes then it has to be encapsulated in multiple CAN frames. In one transmission, there might be a huge load of response data sent from the ECU, which may cause the bus to overload. Consequently, the flow control method is a countermeasure to such a circumstance. Figure 3, illustrates the flow chart of the transport layer that was created to support all of ISO-15765 frame types.

C. Application Layer Development

The UDS Protocol resides in the application layer. It processes data that is stored in message buffers. The protocol will process only a completely received message in a buffer to ensure the correctness of the request. Therefore, a UDS function can only be called by the use of a Single Frame and the last Consecutive Frame. The server must then react to the request and send a response message following the defined standard of ISO standard or SAE standard. The development of application layer is done by completing each service of UDS protocol following a specification of ISO standard and SAE standard. As said, the standard provides a specification of services, request messages and response messages. Each service of UDS protocol can be developed differently as long as the format of request messages and response messages are the same as standard specified.

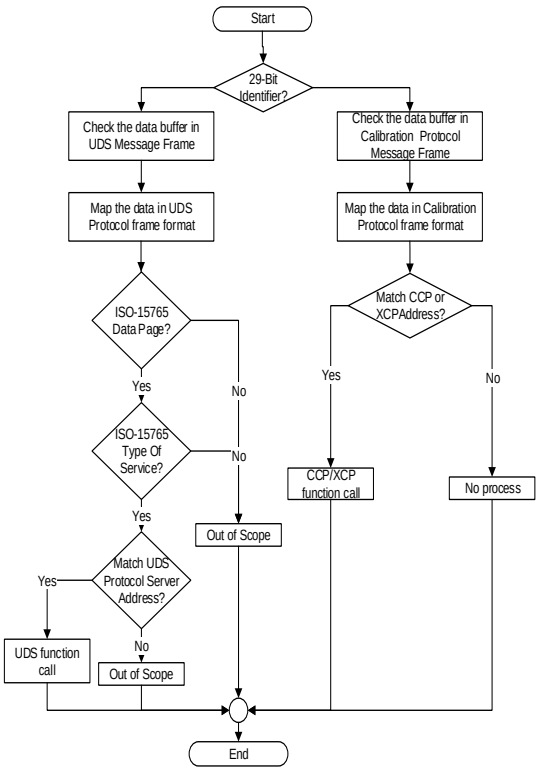

Figure 2: Network Layer Development

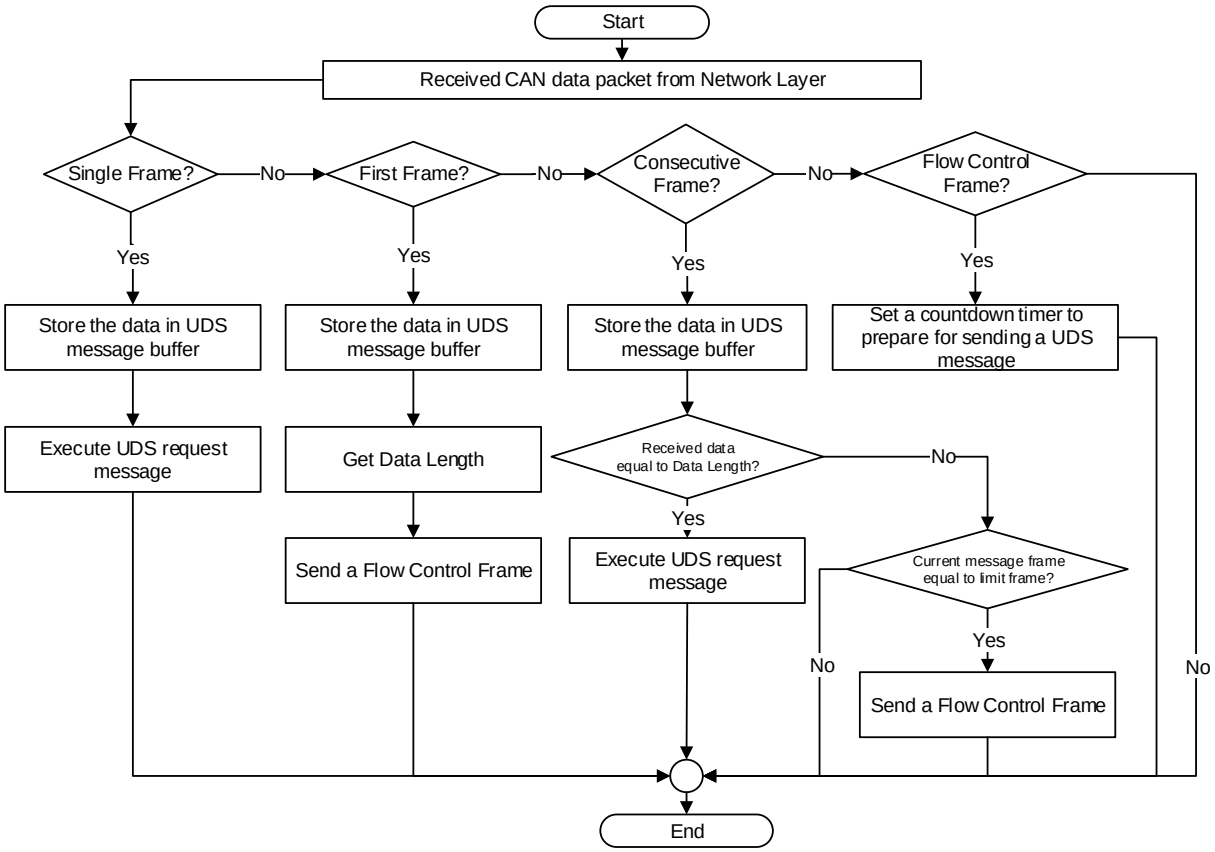

Figure 3: Transport Layer Development

IV. RESULT AND FUTURE WORK

From the concept in above section, the author managed to implement the UDS protocol in the ECU successfully. The correct function of the Network Layer developed was verified by observing CAN IDs between request messages and response messages. Figure 4, is an example of the CAN identifiers between a request message from a client and a response message from a server (ECU). The Transport Layer was checked by generating 4 types of communication. The type of communication consists of Single Frame request-Single Frame response, Multiple Frames request-Single Frame response, Single Frame request-Multiple Frames response and Multiple Frame request-Single Frame response. Figure 5, schematically shows a communication flow for each type of communication. Finally, the Application Layer development was verified by comparing results of the request messages and response messages with the Standard specification for each service. Since the UDS standard provides a large number of services, each of them defining lots of protocol details, a presentation and discussion of the application layer test results would go beyond the scope of this article.

In the research phase, the UDS Protocol has been implemented based on the OSI model. An implementation of Network and Transport Layers has been done, but it still needs further improvement, which comprise error handling and minor functionalities which are described in the following list:

- The Network Layer needs the ability to manage the priority of each CAN message.
- The Application Layer was the most complex task. The following services have been implemented and verified successfully:
    a) DiagnosticSessionControl
    b) ReadMemoryByAddress
    c) WriteMemoryByAddress
    d) SecurityAccess
    e) TransferRequest
    f) TransferExit
    g) RequestUpload
    h) RequestDownload
- The remaining 16 services of UDS protocol need to be implemented in the future.

Last but not least, an approach of the UDS protocol that specific by a standard currently exist only in a CAN bus that has 29-bit frame. As a result, processors that have the right to implement such a protocol, must have a capability of CAN extended format frame (CAN 2.0 B).

29 bits CAN identifier																													
28 26			25	24	23	22	21 11														10 0								
Priority 0x5			ISO 15765 Format		Type of service ISO 15765-3 messages		Source Address 0x1F0										Destination Address 0x1F8												
1	0	1	1	1	1	1	0	0	**1**	1	1	1	1	0	0	0	**0**	0	0	1	1	1	1	1	1	0	0	0

29 bit CAN ID - Client requests

29 bits CAN identifier																													
28 26			25	24	23	22	21 11														10 0								
Priority 0x5			ISO 15765 Format		Type of service ISO 15765-3 messages		Source Address 0x1F8										Destination Address 0x1F0												
1	0	1	1	1	1	1	**0**	0	1	1	**1**	1	1	**1**	0	0	0	**0**	0	1	1	**1**	1	1	0	0	0	0

29 bit CAN ID - Server Response

Figure 4: CAN identifier test for the Network Layer Development

ACKNOWLEDGMENT

Firstly, I would like to acknowledge the following people for their support and assistance with the research. From Ricardo Deutschland GmbH Aachen Technical Centre, the special thank goes to my helpful advisors, Dipl.-Ing. Daniel Gross and Dipl.-Ing. Christoph Lapp-Emden. The advices and support that they gave truly helped the progression and smoothness of the research. The cooperation was much indeed appreciated. The project during the program would be nothing without the enthusiasm and imagination from both of you. Besides, the research made me realize the value of working hard and the challenges to acquire a new knowledge which can encountering us in every minute. Also, my grateful thank to Dr.-Ing. Michael Heeg, my team leader in the company, who always supervised me during the research period at Ricardo. Next persons are my supervisor and my advisor from RWTH Aachen University, Prof. Dr.-Ing. Stefan Kowalewski and Dipl.-Ing. André Stollenwerk, respectively. I am truly grateful for the chance and coordination between RWTH and KMUTNB-TGGS that made this master thesis program possible. The last but not least is Dr. Kamol Limtanyakul, advisor from KMUTNB-TGGS who encouraged and guided me during the research period.

REFERENCES

[1] H. Kleinknecht. CAN Calibration Protocol Version 2.1. Technical report, Standardization of Application/Calibration Systems task force, 1999.

[2] ISO 14229. Road vehicles - Unified Diagnostic Services (UDS). ISO, Geneva, Switzerland 2006.

[3] ISO 14230. Road vehicles - Diagnostic Systems - Keyword Protocol 2000. ISO, Geneva, Switzerland 2006.

[4] ISO 15765. Road vehicles - Diagnostics on Controller Area Networks (CAN). ISO, Geneva, Switzerland 2006.

[5] ISO 15031. Road vehicles – Communication between vehicle and external equipment for emissions-related diagnostics. ISO, Geneva, Switzerland 2006.

[6] SAE J1979. E/E Diagnostic Test Modes. SAE International, 2002.

[7] Roel Schuermans. XCP Version 1.1. Technical report, Association for Standardisation of Automation and Measuring Systems, 2008.

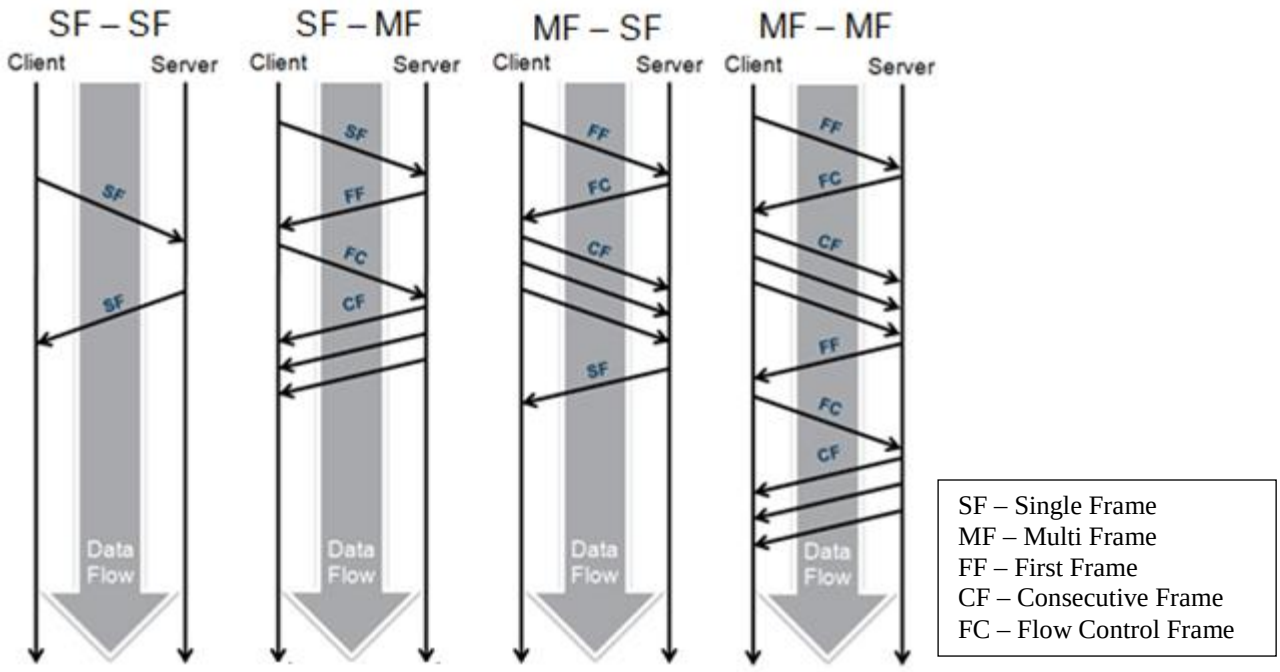

Figure 5: Transport Layer Development - 4 types of Communication