Simulator for the Automotive Diagnosis System on CAN using Vector CANoe Environment

Razvan G. Lazar and Constantin F. Caruntu
Department of Automatic Control and Applied Informatics,
Gheorghe Asachi Technical University of Iasi, Iasi, Romania
Email: lazar.razvan@ac.tuiasi.ro

***Abstract*—One of the current hot topics, which presents a constant development achieved at a very high pace, is the automotive domain. Nowadays, this industry has reached the peak period of its rise, and yet we are increasingly aware that with each passing day this industry leaves more room for development and innovation. In this paper, a Vector CANoe simulator was developed to implement the diagnosis system between two controller area network (CAN) nodes. The architecture is based on two CAN networks, with two nodes connected to each. For the development of the diagnosis system, several functions have been created for sending and receiving messages, but also the ISO-TP and UDS protocols have been implemented. The obtained results illustrate that the requests and responses are in accordance with the implemented protocols and meet the desired requirements.**

***Index Terms*—Automotive diagnosis, Controller Area Network, CANoe environment, ISO-TP protocol, UDS protocol**

I. INTRODUCTION

If the technologies that emerged over the years were thoroughly analyzed, the industry that would capture our attention the most through its major importance development, is the automotive industry. Case studies show that, over the years, this industry had a constant evolution of the used technologies, and yet, it can be said that it leaves more and more room for innovation and development. The trends in the evolution of the automotive industry illustrate that innovation and development are on the rise, but especially for the following topics: electrification of vehicles, connectivity between automotive systems, but also connectivity of the vehicle with certain external databases / external systems, as well as on the subject of autonomous management. Among the secondary development trends - here one can think about a branch of the development for the connectivity in automotive systems - a topic of increased interest is related to the diagnostics of automotive systems [1]–[5].

With the progress of automotive fault diagnosis technology, the involved criteria were gradually improved and unified. Nowadays, the application of diagnosis standard is not only limited to faults, but also in the whole life cycle of vehicle development, test, production and after-sale maintenance. Moreover, the diagnosis system can be extended from its original function to perform the following: calibration, test electronic control units (ECUs), diagnose faults, measure parameters and upgrade code [1]–[5]. In [6], an overview of the functional concepts of different protocols used in vehicle networks is provided. With the help of the Vector CANoe tool [7] all protocols were tested and diagnosed using the three-phase development process. In [8], the UDS [9] Security Access service (0x27) is detailed and tested in practice. Furthermore, the utility of gateways in automotive communication systems have been thoroughly studied in [3], [6], [10]–[12].

This paper proposes the means to simulate the sending and receiving of diagnostic data between two CAN nodes, located on different CAN networks, connected through a Gateway node. The simulation was performed using the CANoe environment [7], which can realize the link between the virtual bus and the real physical bus through Vector's CAN bus interface hardware [13]. The complete digital simulation of the bus application system based on completely virtual nodes can be realized using CANoe. Also, the half-physical simulation of the combination of physical nodes and virtual nodes can be realized, along with the real-time monitoring of the real physical bus communication. CANoe supports the whole process of bus development - from the initial design, simulation and final test analysis, and realizes the seamless integration of the design, simulation and test of the network. Thus, in this paper, the two CAN networks and their associated nodes were created using CANoe. Moreover, several functions necessary to perform the diagnosis: sending messages and receiving messages on CAN [13], based on the ISO-TP [14] and UDS [9] protocols, were also implemented. Compared to [8], in this paper, two other UDS services, i.e., 0x22 and 0x2E, were approached and tested.

The rest of the paper is organized as follows. Section II deals with the communication between automotive subsystems, describing the used protocols, the way of sending data to the ISO-OSI stack and the role of a gateway node. The third section presents the diagnosis in the automotive field, by describing in detail the protocols used for diagnosis. In section IV, the various results obtained for several diagnostic scenarios are described. Section V presents the conclusions and future directions of research.

II. THE AUTOMOTIVE COMMUNICATION SYSTEM

Nowadays, the vehicles become increasingly connected, and also progressively able to control some driving functions, the concept reaching the point in which it is desired that vehicles would have the ability to drive themselves. With the advent of these changes, it can be observed that

*Part of this work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS/CCCDI UEFISCDI, project number PN-III-P2-2.1-PTE-2019-0731, within PNCDI III.

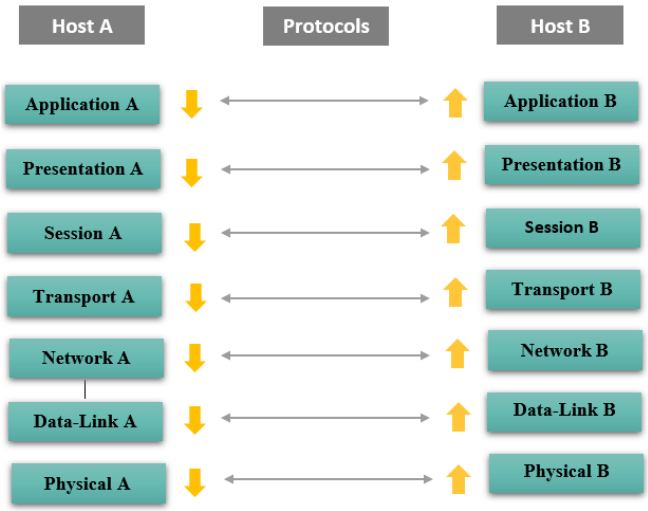

Fig. 1. Data-packet transmission according to the ISO-OSI model

vehicles have come to deal with an increasing volume of data that needs to be retrieved, and then processed, from a very large number of sensors. The number of nodes, e.g., sensors, actuators, entertainment and navigation systems and their corresponding ECUs has been growing exponentially in the typical vehicle. Thus, digital devices and systems must communicate via electrical or optical signals employing well-defined protocols, which, in the automotive industry, are mainly classified according to the speed of data transmission. The Society of Automotive Engineers (SAE) proposes the following classification of communication systems/networks: *i)* Local Interconnect Network (LIN) - actuation of mirrors, electric windows; *ii)* Controller Area Network (CAN): "low speed" - on board instruments; "high speed" - engine management, transmission, braking systems (ABS); *iii)* FlexRay and Ethernet - "X-by-wire" systems, multimedia. Also, an increase in data traffic can be observed between the existing ECUs in vehicles, but also the tendency to decrease the number of ECUs in a vehicle. Thus, the Gateway concept was introduced in the automotive industry to route the exchange of information between ECUs that are connected to different communication networks.

A. *Information sharing*

In automotive systems, the exchange of information is carried out in the same way as in the case of the classical Internet, with the difference that, depending on the requirements of the protocols used and the needs of the developed applications, the ISO-OSI model can be redefined using only part of the levels from the original ISO-OSI model. The way of transmitting the information according to this model is presented in Fig. 1 and consists of the following components: physical level, data link level, network level, transport level, session level, presentation level, and application level, each of which requiring certain specific communication protocols.

The packet of useful information (excluding the metadata added by the protocol) to be transmitted depends on the used communication protocol. Thus, in the case of CAN or LIN protocols, only 8 bytes of useful data can be transmitted compared to the CAN-FD protocol, which can transmit up to 64 bytes of useful data, while for Ethernet, the packet of useful data can reach up to 1518 bytes.

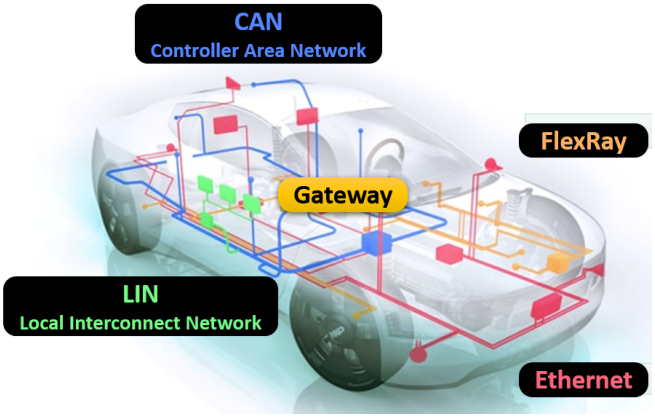

Fig. 2. The role of a gateway in a vehicle

B. *The role of a Gateway in an automotive system*

Apart from multiplexing the communication and information exchange between ECUs, due to the development of this field, the introduction of the concept of gateway in automotive systems (see Fig. 2) is of particular importance. In the automotive field, the gateway ECU includes a number of functions, among which one can mention: diagnosis (both via CAN and Ethernet interfaces), software update in memory, message mirroring on the diagnostic bus, network management, but also a more special function called data routing. This function includes a number of sub-functions such as: message routing, packet data routing, signal routing (sometimes it is necessary to process signal values), high priority routing and diagnostic routing [10].

The data received / sent by a gateway is edited in the form of protocol data units (PDUs), and then transmitted on the communication buses of the vehicle using messages. These messages are transmitted through a physical channel, which in turn presents certain data communication protocols. These include: CAN, LIN, Flex-Ray, MOST, Bluetooth and Ethernet. It should be also specified that a gateway allows the exchange of information similar to a network router. In other words, the routing of information between the ECUs of the vehicle is done through this component.

III. DIAGNOSIS IN AN AUTOMOTIVE SYSTEM

Diagnosis is the ability of any automotive system to check for any problems that may have an impact on the vehicle, to check and then code/decode certain functions of the automotive system, to read diagnostic trouble codes (DTCs), to read in real-time various parameters (such as engine temperature, engine speed, vehicle speed, etc.), or other functions performed by diagnostics.

When discussing the subject of diagnosis in an automotive system, one of the important aspects that must be emphasized is that only six levels of the ISO-OSI model are relevant, which is detailed in Table I.

TABLE I
PROTOCOLS USED IN DIAGNOSIS ACCORDING TO THE ISO-OSI MODEL

ISO OSI Stack Levels	Communication Protocol
7. Application	UDS (ISO-14229)
6. Presentation	-
5. Session	ISO-TP (ISO-15765)
4. Transport	ISO-TP (ISO-15765)
3. Network	ISO-TP (ISO-15765)
2. Data Link	CAN (ISO-11898)
1. Physical	CAN (ISO-11898)

In the tested case, the transmission of information will be done through the CAN protocol. This protocol uses only the first two levels of the ISO-OSI model: the physical level (with the role of transmitting the useful data-packet through the physical communication channel, in this case the two twisted wires) and the data link level (with the role of providing data transport as secure as possible, but also reliably along the physical communication channel). However, the diagnosis of automotive systems also requires the use of network levels (to achieve the best possible transfer of information, by fragmenting and reassembling the data package), transport (to achieve the transport of information in a way as between the two ECUs), and session (with the role of managing the communication session, such as maintaining the transfer of information between the transmitting ECU and the receiving ECU).

The length of the useful data to be transmitted may be greater than the maximum size of the data-packet that can be transmitted through the CAN protocol, i.e., the maximum length of the useful data packet that can be transmitted via the CAN protocol is of 8 bytes. Therefore, in the case of automotive systems, a specific transport protocol named ISO-TP can be used [14]. It has the role of segmenting the useful data-packet that includes the metadata introduced by this protocol into 8-byte data blocks, and then transmit it to the diagnostic bus.

The diagnosis of automotive systems also requires the last level found in the ISO-OSI model, i.e., the application level, with the role of sharing resources and accessing files. In the automotive industry, this level is implemented through the UDS protocol [9], which will be also used in the application defined in this paper.

A. ISO-TP Protocol

The ISO-TP protocol, also called "Transport Layer", is a standard found among the standardized protocols by the International Organization for Standardization (ISO), which is used in the diagnosis of automotive systems for sending information packets (called frames or messages) via the European On-Board Diagnostics (EOBD) connector to the CAN bus. The classic CAN protocol allows the transmission of data packets of up to 8 bytes. Due to the fact that the useful information packets to be transmitted have a variable length, and in many cases their length exceeds the maximum length allowed by the CAN protocol, this protocol helps by fragmenting the initial information packet into smaller data-packets; also, some metadata is added. These packets, which have been created from the fragmented data and the metadata, will be transmitted to the receiving ECU. Due to the information added by the ISO-TP protocol, i.e., metadata, the fragmented data packets can be reconstituted into the initial information packet. An important aspect that needs to be specified is the fact that through this protocol an information packet with a maximum length of 4095 bytes can be transmitted. The types of transmission using the ISO-TP protocol are detailed in Table II.

TABLE II
TYPES OF TRANSMISSION USING THE ISO-TP PROTOCOL

Transmission type	Bits 7-4 Byte 0	Bits 3-0 Byte 0	Bits 15-7 Byte 1	Bits 23-16 Byte 2	Bits 63-24 Byte 3-7
Single frame	0	length (0-7)	maximum 7 bytes of useful data		
First frame	1	length (8-4095)		6 bytes of useful data	
Consecutive frame	2	index (0-15)	up to 7 bytes of useful data		
Flow control	3	Flag (0,1,2)	Block size	ST time	Pending

TABLE III
GENERIC STRUCTURE OF MESSAGES ACCORDING TO THE UDS PROTOCOL

Request made by sender				
SID (1 byte)		Parameters (n bytes)		
Type of request made by receiver				
a) Positive response		**b) Negative response**		
SID + 0x40 (1 byte)	Mirroring the main parameters (n bytes)	Identifier 0x7F (1 byte)	Mirroring SID (1 byte)	NRC (1 byte)

B. UDS Protocol

To cover the entire redefined ISO-OSI stack, the diagnostics of automotive systems uses the Unified Diagnostic Services (UDS) communication protocol for the application level. This protocol can be implemented over the KWP2000 (ISO-14230) and ISO-TP (ISO-15765) protocols, among the ISO standardized protocols.

The design of a requirement that is achieved through the UDS protocol must contain two components: service identifier (SID - service ID) and parameters specific to the requested service (see Table III for details). For such a request, two types of responses could be given: positive response or negative response. The structure of a positive response consists of two components: the mirroring of the SID which adds the value 0x40 (hexadecimal value) and the mirroring of the main parameters that depend on the requested service. The structure of a negative response consists in turn of three components: the Identifier field specifying the response for a negative message (hexadecimal value 0x7F) to the received request, the mirroring of the requested service ID (SID) and the field NRC (negative response), which has the role of specifying the reason for sending a negative response.

IV. SIMULATION RESULTS

The Vector CANoe environment was used to perform the simulation, implement the necessary functions and analyze

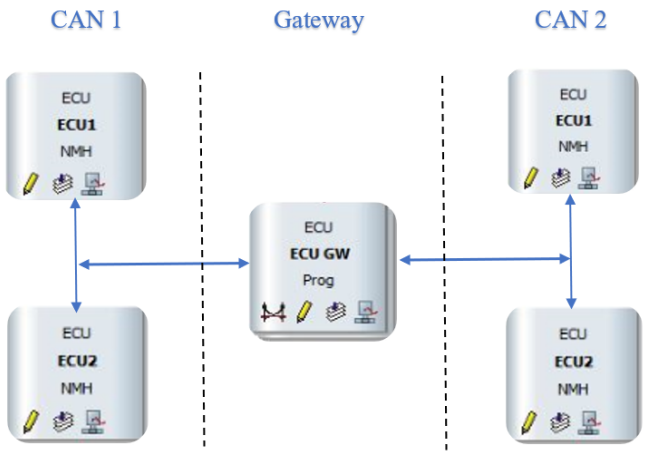

Fig. 3. System architecture

the results. CANoe is a software development and testing tool mainly used by vehicle manufacturers and suppliers of electronic control units for the development, analysis, simulation, testing, diagnosis and start-up of communication networks and ECUs. CANoe's simulation and testing facilities are made using the Communication Access Programming Language (CAPL). This programming language is event-oriented, which can be of several kinds, e.g., there are events for receiving a message, setting a signal, starting/stopping the simulation, setting variables, etc. In addition to these events, the way the code is written is very similar to the C language. This environment can support several communication protocols: CAN, LIN, FlexRay, Ethernet and MOST; as well as the CAN-based protocols: J1939, CANopen [7].

A. System architecture

To perform a simulation within the CANoe application some input files (databases) are needed, in which various information about the communication networks can be collected, e.g., the ECUs on each network, the messages sent by each ECU and the signals related to each message. Depending on the needs, a configuration has to be created in which the CAN networks and ECUs can be found. The developed application uses two simulated CAN networks, connected through a gateway node, with which the CAN messages can be simulated and the code to send the diagnostic request between the two nodes can be developed (see Fig. 3).

B. Diagnostic implementation architecture

In carrying out this implementation, both CAN networks were used, as the purpose is to read the information from ECU2 on CAN2, using a request sent from CAN1 (see Fig. 4). The DiagRequestCAN1 request is transmitted from CAN1 using the ECU1, further routed by the gateway (GW) node into DiagRequestCAN2. The latter reaches the CAN2 network, and then the ECU2, which responds with the message DiagRespCAN2.

1) Forming and sending the request: To facilitate the testing of use-cases, an interface was created through which the type and parameters of the request, but also of the response can be configured (see Fig. 5).

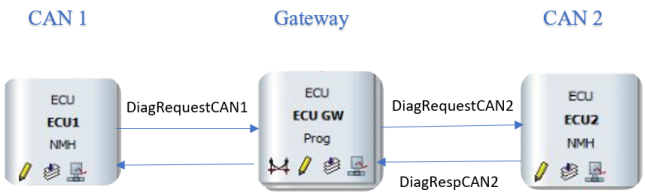

Fig. 4. Diagnosis implementation architecture

To send the message, each section of the "CAN1 - Input Message" interface must be configured:

- Message Parameters → ID: 0x700 - message identifier; Channel: CAN1 - the channel on which the message is sent; Data Length: 8 - number of data bytes;
- Data Field → the value for each byte of the message must be given, starting from the left side with byte 0 to byte 7; also, the type of request, i.e., writing (0x2E) or reading (0x22), must be chosen;
- Triggering → Cycle Time: 0 - the cyclicity of the message, which is set to 0 in our case because the diagnosis does not use cyclic messages; Send once: spontaneous sending of the message, only once; Send cyclic: send the message cyclically.

2) Request routing: The message is routed by the ECU GW node that appears on both CAN networks. Once the message is transmitted by ECU1 from CAN1, ECU GW receives this message and copies its data inside the *DiagRequestCAN2* message, which will be sent on CAN 2 to ECU2 with the ID 0x703.

3) Forming and sending the response: Using the interface described above, one can choose the desired response, according to the UDS protocol, and depending on the length of the data entered in the Data Field, the transmission type will be chosen according to the ISO-TP protocol. The following cases should be differentiated: *i)* if a read request is received, the exact number of bytes must be calculated to know precisely what type of transmission will be used: if the length is less than 6, then Single Frame will be used, otherwise the First Frame together with the Consecutive Frame will be used; *ii)* if a write request is received, the time at which a flow control message should be sent and the type of response should be considered. As mentioned in subsection III-B, for such a request, two types of responses could be given: positive response or negative response.

C. Illustrative results

In what follows, several test scenarios defined to test the developed simulator are briefly described. The graphical interface of the simulator is illustrated in Fig. 5, in which one can observe the "CAN1 - Input Message" fields and the "CAN2 - Slave diagnostic" options.

*1) Request - Read (*0x22): Using the graphical interface shown in Fig. 5, the user can choose the wanted type of answer and by writing the data field for the response, as follows:

a. Negative Response (NRC = 0x0A) using a Single Frame (data < 6), illustrated in Fig. 6;

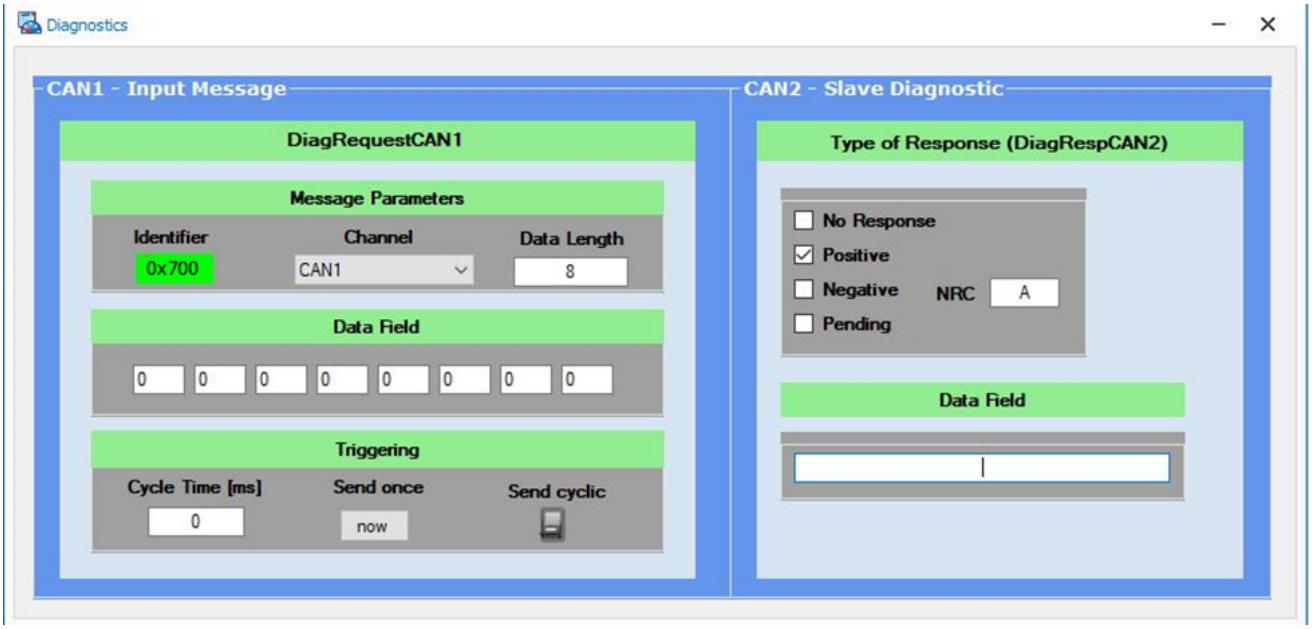

Fig. 5. Graphical interface for simulating the diagnosis system

b. Positive Response using a First Frame and 3 Consecutive Frames (data $>$ 6 bytes), represented in Fig. 7;

c. Pending Response - NRC = 0x78 using 3 Single Frames (3 pending messages), illustrated in Fig. 8;

d. No Response (ECU2 will not respond to the request), represented in Fig. 9.

*2) Request - Write (*0x2E): For the write request, in the interface shown in Fig. 5, only the type of response, positive or negative, must be selected:

a. Positive Response using the Single Frame, represented in Fig. 10;

b. Negative Response (NRC = 0x12) using the Single Frame, illustrated in Fig. 11.

V. CONCLUSION AND FUTURE WORK

In this paper, a simulator was developed for the automotive diagnosis system on CAN using the Vector CANoe environment. Two CAN networks were implemented along with two nodes connected for each of them. The obtained results illustrated the capabilities of the proposed simulator in terms of requesting responses and responding to requests, and the requirements were met in accordance with the implemented transmission protocols. The proposed application can fold very well for the following purposes: *i)* testing the communication on CAN between the developed simulator and any ECU on a real vehicle; *ii)* testing the ability of a real ECU to send a diagnostic request or to respond to a diagnostic request sent from the performed simulation; *iii)* transposing the developed simulator in the form of a learning platform for new employees in the automotive field. Future work will focus on extending this simulator by diversifying the used communication protocols, e.g., LIN, CAN FD, ETHERNET, and on how to implement the diagnosis for each of them.

REFERENCES

[1] J. Suwatthikul, R. McMurran, and R. P. Jones, "Automotive network diagnostic systems," in *International Symposium on Industrial Embedded Systems*, 2006.

[2] P. Peti, A. Timmerberg, T. Pfeffer, S. Muller, and C. Ratz, "A quantitative study on automatic validation of the diagnostic services of electronic control units," in *IEEE International Conference on Emerging Technologies and Factory Automation*, 2008.

[3] Y. S. Lee, J. H. Kim, and J. W. Jeon, "Automotive diagnostic gateway using diagnostic over Internet protocol," *IEIE Transactions on Smart Processing and Computing*, vol. 3, no. 5, pp. 313–318, 2014.

[4] A. D. Sutar and S. B. Shinde, "ECU diagnostics validator using CANUSB," in *International Conference on Inventive Computing and Informatics*, 2017.

[5] B. Prasad, J.-J. Tang, and S.-J. Luo, "Design and implementation of SAE J1939 vehicle diagnostics system," in *IEEE International Conference on Computation, Communication and Engineering*, 2019.

[6] R. Rishvanth, D. Valli, and K. Ganesan, "Design of an in-vehicle network (using LIN, CAN and FlexRay), gateway and its diagnostics using Vector CANoe," *American Journal of Signal Processing*, vol. 1, no. 2, pp. 40–45, 2012.

[7] Vector Informatik GmbH, *CANoe - User Manual*, 7th ed., Vector, Stuttgart, Germany.

[8] D. Hu, D. Hou, K. Guo, and C. Sun, "Design and implementation of diagnostic system for integrated body controller based on CAN bus," in *Chinese Automation Congress*, 2019.

[9] "Road vehicles - unified diagnostic services (UDS) - specification and requirements," International Organization for Standardization, Geneva, CH, Standard, Mar. 2013.

[10] S. Seo, J. Kim, T. Moon, S. Hwang, K. Kwon, and J. Jeon, "A reliable gateway for in-vehicle networks," *IFAC Proceedings Volumes*, vol. 41, no. 2, p. 1208112086, 2008.

[11] S.-H. Kim, S.-H. Seo, J.-H. Kim, T.-M. Moon, C.-W. Son, S.-H. Hwang, and J. W. Jeon, "A gateway system for an automotive system: LIN, CAN, and FlexRay," in *IEEE International Conference on Industrial Informatics*, 2008.

[12] Z. Rui, Q. Gui-he, and L. Jia-qiao, "Gateway system for CAN and FlexRay in automotive ECU networks," in *International Conference on Information, Networking and Automation*, 2010.

[13] "Road vehicles - controller area network (CAN) - Part 1: Data link layer and physical signalling," International Organization for Standardization, Geneva, CH, Standard, Dec. 2015.

[14] "Road vehicles - diagnostic communication over Controller Area Network (DoCAN) - Part 2: Transport protocol and network layer services," International Organization for Standardization, Geneva, CH, Standard, Apr. 2016.

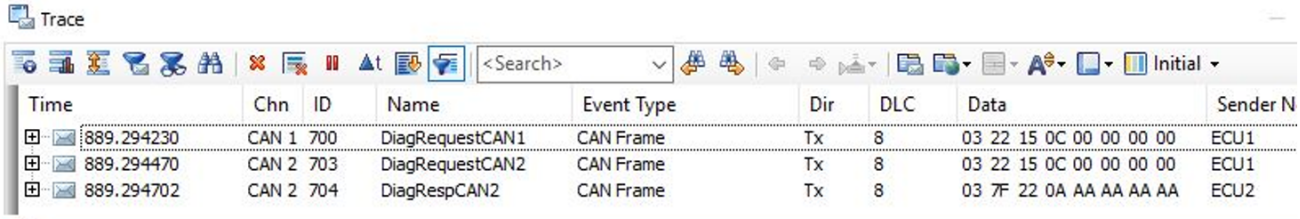

Fig. 6. Read: Negative response

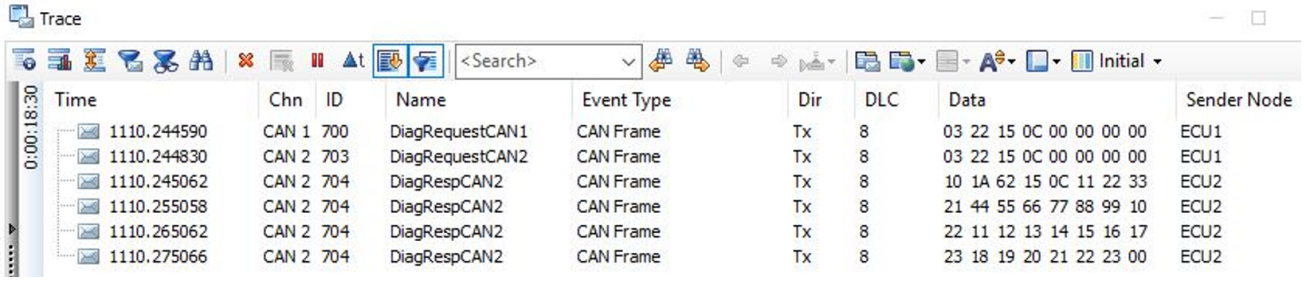

Fig. 7. Read: Positive response

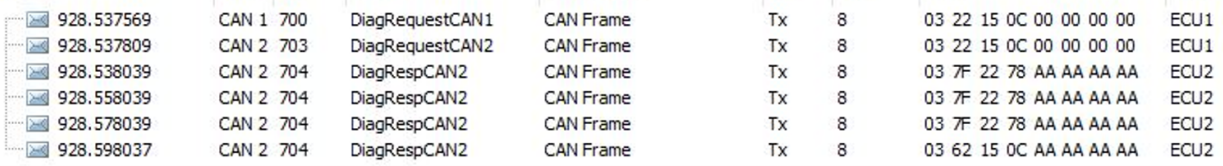

Fig. 8. Read: Positive pending response

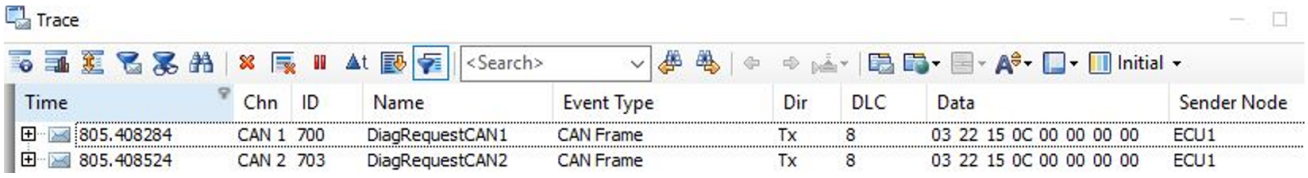

Fig. 9. Read: No response

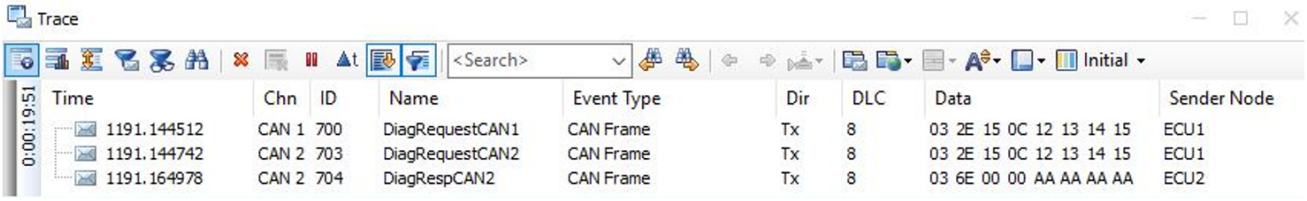

Fig. 10. Write: Positive response

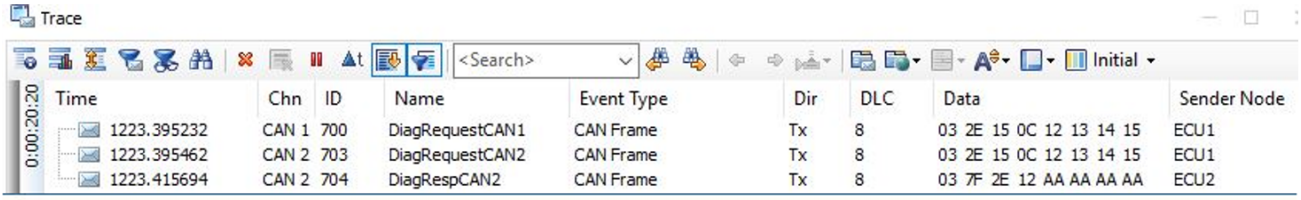

Fig. 11. Write: Negative response