Implementation of Read DID UDS Service based on AUTOSAR

MADAN MOHAN N
MTech Student in VLSI Design and Embedded System, School of Electronics and Communication, REVA University
Bangalore India

Dr. BHARATHI S H
Professor School of Electronics and Communication, REVA University
Bangalore India

***Abstract*—**The AUTOSAR is a standard platform on which the future vehicle applications are implemented and it minimizes the current barriers between functional domains. The unified diagnostic services (UDS) based on AUTOSAR software architecture is implemented. The work include implementation of one of the UDS Service Read Data By Identifier and reading multiple Data Identifier(DID) in a single request which is provided in DCM(Diagnostic Communication Manager).And Implemented service is tested using CANoe.

Keywords*— *UDS, DCM, Data Identifier,AUTOSAR, CANoe

I. Introduction

The Automotive Industry due to stringent legislative emission norms and due to lot of competition, embedded system took major role in automotive control this embedded system were called ECU(Electronic control unit) for example Engine ECU have process control over engine functionalities like exhaust system, injection system, ABS ECU to have control of braking system ,Radar and entertainment ECU for entertainment purpose and so on. Today in an average car more than thirty five ECU's will be present. These ECU are embedded system means hardware and software to release the functionalities which is intended to do. Initially each company use to have their own software architecture for their ECU, but now a days all the car manufacturer follows single software architecture foe their ECU's called as AUTOSAR. AUTOSAR stands for Automotive Open Software Architecture. Depending on different functionalities ECU software consist of different packages for example COM stack handles inter ECU communication . Similarly tester related communication handled by AUTOSAR package called as DCM (Diagnostic communication manager). An external tester tool is connected to ECU in order to retrieve some information from ECU or to write something on ECU or to trigger some events on ECU , this happens during the UDS protocol in service request and response format . In each ECU particular software intended for tester ECU package ,DSP takes request from the tester , DSD process the request and accumulate the response and sends back the proper response to tester. These functions are done by the module called DCM. In Diagnostic module contains different service. This paper deals with implementation of Diagnostic Read Data Identifier Service. Generally tester are interested in reading multiple Data Identifier at a time but CAN communication protocol supports only 8bytes of data to request. So this paper gives idea of requesting multiple DID's in a single request through CAN communication protocol.

II. CAN DIAGNOSTIC STACK IN AUTOSAR

A. Diagnostic Stack and Communication Path

Tester and DCM happens via communication stack part which implements a communication protocol like Ethernet ,FlexRay, CAN etc. In this paper we have used CAN for Communication protocol . So this DCM along with part of communication stack together is called Diag Stack DCM handles service protocol like UDS, remaining layers handles communication protocol like CAN . When we relate the DiagStack layers with the OSI model layers then Can driver layer represents physical layer ,then CAN Interface layer represent data link layer , then CanTp represents Transport layer, PduR layer represent network layer and DCM layer represents application and session layer . If the Diag Stack is explained in OSI model terminology then each layer excepts data packet from the upper layer called Service Data Unit (SDU) and then adds its own protocol control information(PCI) to SDU and forms Protocol Data Unit (PDU) and send it down to further lower layer. This PDU becomes SDU for the lower layer and lower layer adds its own PCI to make its own PDU and so on. From this it clears that PDU is different for each module of the DiagStack. For DCM module it's called I-PDU which is complete service request message or response message this I-PDU is present in DCM buffer. In PduR module it is also called I-PDU its just routes the PDU without any modification. In CanTP module this I-PDU will be seen as N-SDU. CanTP handles segmentation of huge data into smaller chunks fit for data length of underlined communication protocol. CanTp breaks the service message which is same as N-SDU into smaller chunks and adds its own N-PCI as defined. And forms N-PDU. This N-PDU act as Data field for underlying communication protocol which is CAN. N-PDU can be maximum of 8 bytes. One N-SDU is broken into multiple N-PDU which will be converted into multiple frames for lower layers. CanTp module handles multi frame communication in AUTOSAR. This N-PDU moves to Data Link layer or CanIf(Can Interface) and acts as L-PDU and gets its PCI to become its L-PDU.L-PDU is consist of CAN message Id , Data Length Code, and the actual data .And this L-PDU is transferred to tester through CAN bus

III. FLOW CONTROL

A. *Flow Control 1*

It is important to know about flow control frames because the Read Data Identifier service can be used to read multiple DID at a time. But as a known fact CAN communication protocol cannot handle more than 8bytes.So in ordered to make request the service with multiple Data Identifier Flow control frame is used. Flow control is a mechanism of communicating huge chunks of data between ECU and tester by means of single frame or multiple frame. This mechanism is described in ISO15765 protocol and handled in CAN-TP module in autosar architecture. There are 4 types in Flow-Control frame they are Single Frame(SF), First Frame(FF), Consecutive Frame(CF) and Flow Control Frame. The sequence of flow control between tester and ECU 1. The size of N-SDU is less than data limit of frame, since we are using CAN data limit is 8 bytes. In this case we use single frame. The transmitter sends entire data in a single frame to the receiver. 2. The size of N-SDU is greater than data limit. In this case multi-frame communication is used. The N-SDU is divided into multiple chunks with each piece after applying N_PCI after forming N-PDU can fit in one frame of underlying communication protocol and transmitted one by one until complete data is transmitted successfully. The transmitter sends first piece as a first frame to the receiver and the receiver responds the flow control frame to the transmitter and then transmitter sends series of consecutive frames one after the other in the correct order to the receiver. Each consecutive frame carries one piece of N-SDU respectively

B. *Flow Control 2*

In a multi-frame communication initiated by the sender with a first frame the receiver needs to tell how the flow of consecutive frames need to be sent to the receiver. There are various parameters to be considered like, does the receiver has enough memory for complete N-SDU and rate at which consecutive frames must be transmitted and so on. All these information needed for controlling flow of frames need to be communicated from the sender to the receiver. Hence the flow control frame. Consider the data field of flow control frame is always 3 bytes. The higher nibble of MSB is index. Index for flow control frame is 3 or 0011 in binary. The lower nibble of MSB is flow status. It can have 3 possible values. They are 0, 1 and 2. 0 represents clear to send the consecutive frames, 1 represents wait for next flow-control frame so the sender does not send any consecutive frame and waits for one more flow control frame from the receiver. And 2 represents abort the multi-frame communication. The 2nd byte of flow control frame is block size. This field is significant only when the flow status value is 0. If FS=0 and BS=0 that means send all the consecutive frames till complete N-SDU is communicated. Receiver makes block size as 0 only when the buffer has a free space to accommodate complete N-SDU. On the other hand if FS=0 and BS=N that means send N consecutive frames and wait for next flow control frame. For example if block size is 5 than it means sender can send 5 consecutive frames and wait for the receiver to send one more flow control frame. This happens when the receiver has one free space in its buffer but not so much as to accommodate complete N-SDU. So depending on how much free space is present it computes how many consecutive frames can be taken in and puts that number as block size. Note that when BS is non zero than then the BS has no significance. Last byte is called ST min which stands for minimum separation time. This field tells what should be the minimum time gap between consecutive frames. Value from 1-127 specify number of milliseconds delay between consecutive frames. Values from 241-249 specify delay increasing from 100ms to 900ms.

The purpose of this Flow Control concept in this research work is because of we are reading multiple DID. Thus Can Communication protocol will handle 8bytes of data at time the request message will contain more than 8bytes of data so this flow control concept is important.

IV. UDS SERVICE

A. *Introduction to UDS*

With rapid implementation of electronic embedded systems in vehicles, the need to track and control vehicles different parameters was important. Thus diagnostic systems were developed so that clients could detect the fault in the vehicle by connecting their diagnostic tester to ECU. But there are so many manufacturers in the world like BMW, Volkswagen, and Ford and so on where each brand will have its own architecture and software's in its ECUs. So it's not possible to make tester communication language for each car brand. In order to make sure that all brands cars can communicate with the single generic testing tool SAE came up with the standard protocol for tester communication. This protocol is named as unified diagnostic services. Every ECU present in every car must be able to communicate with the tester tool as per this protocol. Tester can trigger various actions in the ECU. When I say action it can be requesting data, writing some data, running tests on the car component and getting its result back, flashing a program in ECU, clearing the memory, setting a schedule and many other things. A tester triggers these in the form of service request. Since the tester request for the service it is called a client and the ECU receives the request and provides the service. Hence the ECU is called server. So tester and ECU is present in client-server topology and the UDS is nothing but the collection of diagnostic services which can be requested by the tester as client and performed by ECU as a server. UDS defines the available services, request message, response message, timing parameters and services handled by the tester and ECU. ISO I14229 is a UDS protocol is published in 5 documents where part 1 contains all the details of UDS services.

B. UDS Request Message Format

Request message is always from client to the server. UDS is a collection of diagnostic services. Each service has a service identifier assigned to it. This service identifier is of 1 byte in length. The range of request service id is 0x00 to 0x3E. By this service id the server understands which service is requested by client. For example, if the SID is 0x22 then requested service is read data by identifier or if the SID is 0x31 then the requested service is called routine control. This SID is always the 1st byte of the service request message which is a mandatory field for all the request messages in UDS. The optional field is called sub function whose length is also 1 byte and the value of which tells the server for the requested service which sub functionality is requested. For example for the routine control service (0x31) sub function 0x01 is to start the routine, sub function 0x02 is to stop the routine. Sub function 0x03 is to request routine results. All these service belong to single service called routine control. But to make the server understand what control is needed different sub function are described in the UDS. Sub functions are needed for some selected services only. Services like RDBI, WDBI are also there which does not require sub function. Sub function is an optional field in the service request message. The data identifier field is of 2 byte length. In UDS the client and server only communicate numbers. If you want to read the information engine speed from the tester then you cannot put the string “engine speed” in the requested message. So for all the data elements which the tester may read must be assigned a number already. This number is identifier for that data element. In UDS this identifier number is of 2 bytes length. This number is called data identifier or DID. Let’s say 0x1234 is the number assigned to engine speed. Then both tester and ECU knows that they are talking about engine speed. Likewise different data elements are predefined with data identifiers in both server and client. But this assignment must match with both client and server. So in onboard diagnostics these DIDs are standardized globally. In UDS car manufactures define their own DIDs, only tester tools from the OEMs can read these DIDs.

Service	**Identifier**
Data oriented	DID
Routine control	RID

The purpose of the identifier remains the same. There are services that do not have DID field. So this is also optional field. Single or multiple DIDs can be present in the single service message. Next is data record field is optional as per the service requested. For example read data by identifier does not require data record field in its service request. Whereas, write data by identifier needs it. After mentioning the data identifier, you need to tell the server what data value has to be written to that element. For example the data identifier 0x9876 represents vehicle speed limit and its tester wants to change its current value then you also need to specify what new value has to be stored in data element. Data identifier tells what data element is referred. Data record specifies what new value has to be stored in that data element. In general, data record can be considered as Meta data of DID and it is mandatory for some specific services only.

V. IMPLEMENTATION METHOD *A.*

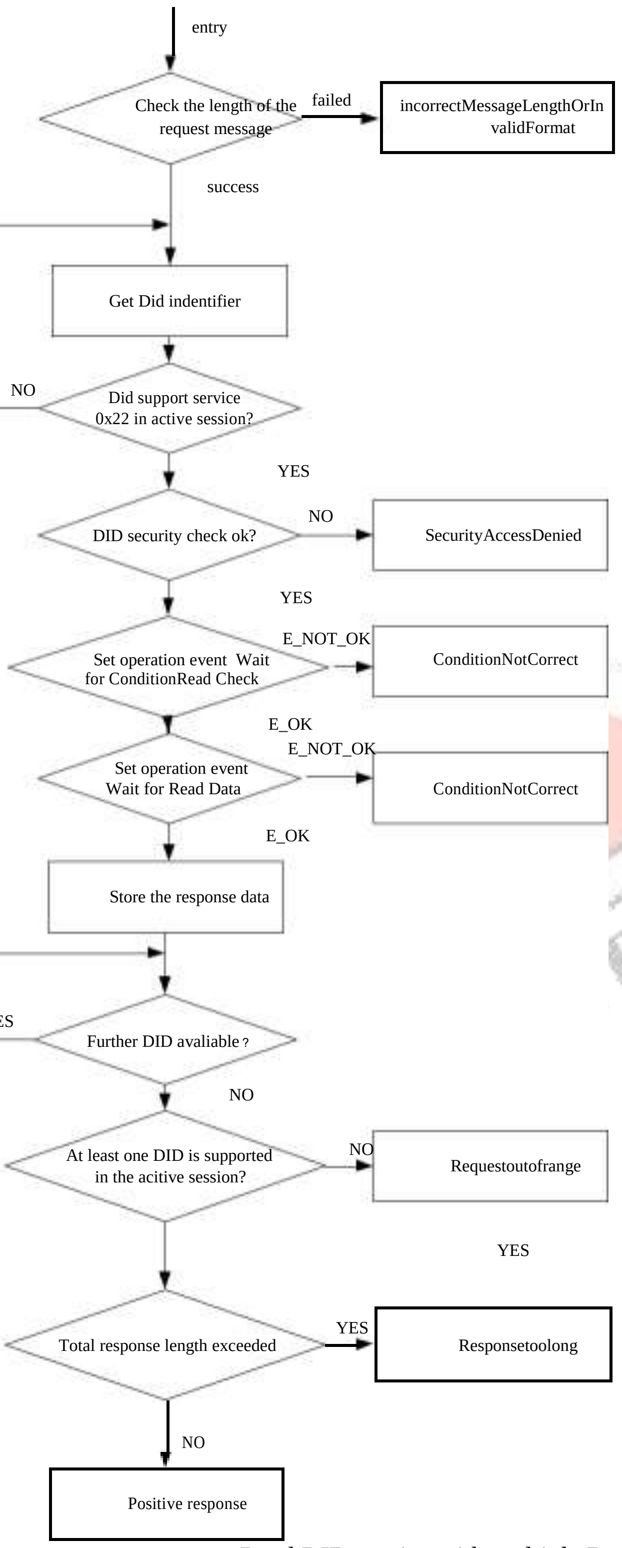
Required Configuration To Test Read DID

The CAN driver module is meant for the hardware access and provides a hardware independent communication to the upper layer. In the Can module configuration Can Controller baudrate is configured , Can transmitting and receiving i.e Tx and Rx processing type is configured. The Configuration of hardware objects like CanHRO with CanID value as 0x72B and its type configured as standard and CanHTO with CanID value as 0x62B and it is also standard type. For remaining communication layer like CanIF, CanTP, PDUR proper Rx and Tx CanID and its type should be mapped and that must be taken care. The DCM module consist of three sub modules those are Diagnostic Session Layer, Diagnostic Service Dispatcher and Diagnostic Service Processing sub module. The DSL will ensures data flow of diagnostic requests and responses, and its take care of diagnostic protocol timing and manages diagnostic states like diagnostic session and security. In DSD , there will be container namely service table in which all the testing services needed to be configured . In this case Read DID service need to be configured with its service identifier value i.e 0x22. The DSP sub module will look into the actual diagnostic service requests. In the DSP sub module all the Did related information is configured, The size of the DID will be two bytes. The Data Identifier is configured with security level and session level.

B. Implemtation

.

Read data by Identifier service . In this paper we are concentrating on the positive response for the requested


entry
Check the length of the request message
failed
incorrectMessageLengthOrInvalidFormat
success
Get Did indentifier
NO
Did support service 0x22 in active session?
YES
DID security check ok?
NO
SecurityAccessDenied
YES
E_NOT_OK
Set operation event Wait for ConditionRead Check
ConditionNotCorrect
E_OK
E_NOT_OK
Set operation event Wait for Read Data
ConditionNotCorrect
E_OK
Store the response data
YES
Further DID avaliable ?
NO
At least one DID is supported in the acitive session?
NO
Requestoutofrange
YES
Total response length exceeded
YES
Responsetoolong
NO
Positive response

Read DID service with multiple Data Identifier which are configured in the DSP container accordingly. To get the positive response for the Read Request these flow chat steps are implemented.

This flow chart [4] will give the clear picture of implementing

VI. RESULTS

At last, we test the Read DID Service by using Canoe. The Test Result is tabulated below. The Read DID service is requested with the Configured multiple DID of value and the received response is validated by making use of Capl Scripts.

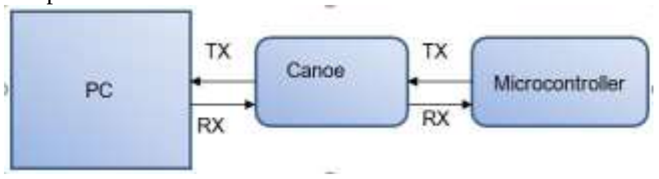


Figure 1. Testing Environment

Channel	Channel ID	Name	Event Type	Direction	Data Length	Data
CAN1	0x72B	Read DID Request	CAN Frame	Tx	8	0422DD0000000000
CAN1	0x62B	Response	CAN Frame	Rx	8	0666DD000A120000

Figure 2.Test Result of Read DID

VII. CONCLUSION

This paper is all about sending the Read DID service Request with multiple number of DID at a time by making use of flow control frame and validating the positive response received from the server. In future this service different NRC can be tested.

VIII.ACKNOWLEDGMENT

I acknowledge our organization REVA university for all support to do this work .

IX. REFERENCE

[1] AUTOSAR Development Partnership. Specification of DCM www.autosar.org

[2] AUTOSAR Development Partnership. Specification of PduR www.autosar.org

[3] AUTOSAR Development Partnership. Specification of CanTp www.autosar.org

[4] XIE, Yue-yin, Z. H. O. U. Chao, and L. U. O. Feng. "Implementation of Automotive Unified Diagnostic Services Based on AUTOSAR." DEStech Transactions on Computer Science and Engineering itme (2017).

[5] Di Natale, M., Zeng, H., Giusto, P., and Ghosal, A. , Understanding and Using the Controller Area Network Communication Protocol, Theory an d Practice, Springer, New York, NY, USA, 2012

[6] ISO15765-3-2004, Road Vehicles—Diagnostics on Controller Area Networks (CAN)-Part 3: Implementation of Unified Diagnostic Services (UDS on CAN), 2004

[7] Jun Jiang, “A network fault diagnostic approach based on a statistical traffic normality prediction algorithm”, Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE,Vol: 5, pp: 2918 - 2922,2003