Circuits and Systems, 2016, 7, 3415-3429
Published Online August 2016 in SciRes. http://www.scirp.org/journal/cs
http://dx.doi.org/10.4236/cs.2016.710291

Scientific Research Publishing

Improved Automotive CAN Protocol Based on Payload Reduction and Selective Bit Stuffing

B. Vinodh Kumar[1], J. Ramesh[2]

[1]Department of Electronics and Communication Engineering, Sri Shakthi Institute of Engineering and Technology, Coimbatore, India
[2]Department of Electronics and Communication Engineering, PSG College of Technology, Coimbatore, India
Email: vinoscholar@gmail.com, jramesh60@yahoo.com

Received 13 May 2016; accepted 23 May 2016; published 31 August 2016

Abstract

In this research paper, the implementation strategies of automotive controller area network protocol are investigated and the short messaging scheme with selective bit stuffing method to improve the effective utilization of its bandwidth has been proposed. There would be a sharp decrease in the performance of traditional CAN protocol because of considerable increase in the number of ECUs (Electronic Control Units) and infotainment gadgets connected in the vehicle architecture. The demand for safety, emission, diagnostics and comfort norms has steeply increased the number of messages in the 250 Kbps CAN network as the computational power of ECUs has gone up. To overcome this problem, the short CAN method has been proposed and the work is benchmarked with SAEJ1939 Heavy commercial vehicle CAN standard. The Matlab Simulink based short CAN has been modeled and the performance of the proposed system has evaluated using virtual instrument cluster. Experimental results have shown that compared to the traditional CAN, the proposed method has reduced the worst case response time of CAN extended frame from 160 µ sec to 144 µ sec. Selective bit stuffing technique has reduced the impact of bit stuffing over the payload and improved the utilization factor for the CAN bus without affecting the CAN message ID properties. The proposed algorithm has been modeled and simulated using CAN Matlab model Simulink and it has been verified using virtual CAN tool and real time CAN bus hardware.

Keywords

CAN, Selective Bit Stuffing, Payload Reduction, Worst Case Response Time

How to cite this paper: Vinodh Kumar, B. and Ramesh, J. (2016) Improved Automotive CAN Protocol Based on Payload Reduction and Selective Bit Stuffing. *Circuits and Systems*, **7**, 3415-3429. http://dx.doi.org/10.4236/cs.2016.710291

1. Introduction

The computational power of the microprocessor and microcontroller keeps on increasing along with the demand of sophisticated vehicle electronics. The complexity of automobile electronics has now become much more complicated as the requirements of vehicle to vehicle data communication systems, navigation systems and other driving assistance systems are steadily growing [1]. In the cost competitive commercial vehicle business, it is challenging for the automobile original equipment manufacturers have a greater responsibility of delivering safe vehicles to sustain world class standards. In this research work, instrument cluster model of a heavy commercial vehicle is being designed using Matlab Simulink and verifies the implementation of short CAN over SAEJ1839 messages with selective bit stuffing. Two virtual Controller Area Network (CAN) nodes have been developed and made communicated with each other as per SAEJ1939 standard. In a basic commercial vehicle, one CAN bus connects all power train related Electronic Control Units (ECUs) such as ABS, Engine control unit, Transmission control units and much more. When it comes to hybrid vehicle or pure electric vehicle, the number of ECUs connected in a vehicle is doubled. Most of the ECUs use CAN as a main communication protocol and it is supposed to be working at 250 Kbps. Research work of Peng Hao and Yang Shun clearly shows that Rate Monotonic based CAN bus network messages will meet all its deadline only when its utilization rate is not above 30% [2]. To overcome this problem, OEMs would always prefer 2 to 3 CAN bus for handling the bandwidth scarcity problem with an increase in the vehicle cost.

Alternate communication protocols like fast CAN, LIN, Flexray and MOST are being considered by vehicle manufactures. But when it comes to cost, performance, reliability, safety, standardization and vendor support CAN outperforms all the other networks. All vehicle network protocols are classified according to their operating speed, length, cost and safety features. Steve C. Talbot and Shangping Ren's work on automotive communication protocols discusses about CAN, LIN and Flexray protocols which are mostly used in vehicle powertrain and safety [3]. LIN is preferred for the applications such as door control, window control, etc. because it supports less than 20 Kbps speed and it is viewed as a sub bus for CAN network. Controller Area Network comes in three speed grades such as basic CAN with 125 Kbps, low speed CAN with 500 Kbps and high speed CAN with 1 Mbps. Depending upon the vehicle architecture requirement, low speed and medium speed CAN are being used. Usually power train CAN is fitted with HS CAN as powertrain handles a higher number of messages and infotainments. CAN is designed with low speed as it has to handle infotainment, information to instrument cluster and the navigation systems. MOST is the special kind of protocol which supports more than 1 Mbps and it is more suitable for audio video communication inside the vehicle

2. CAN History

In 1985, Bosch GmbH had developed CAN for in-vehicle network and then certified with ISO 11,898 international standards. Since then several higher level protocols like CANopen and Devicenet have been standardized for industrial automation. During 1986, Robert Bosch has introduced CAN at Society of Automotive Engineers congress (SAE). Since 1994, several higher-level protocols have been standardized on CAN, such as CANopen and DeviceNet. In February of 1986, Robert Bosch GmbH introduced the serial bus system CAN at the Society of Automotive Engineers (SAE) congress. Since then CAN is being exclusively used in automotive in-vehicle networking. Low cost, reliable, universal standardization, plug and play features with Automatic error correction and detection mechanism makes this CAN protocol as an ideal choice for safety critical applications, but at 250 kbps along with bit stuffing technique for synchronizations and error detection that increases worst case response time and slows the system performance.

2.1. Problem Statement

Many research activities have been carried out for critical time requirements and effective utilization of the available CAN bandwidth. CAN uses Non Return to Zero (NRZ) bit encoding technique ensures that CAN bus does have long sequence of same polarity bits and bit stuffing technique has been used for synchronization of the connected control units connected in the bus. CAN transceiver insert a complimentary logic bit if the 5 consecutive bit sequence is of same value else transmit the original data and at the receiver side these stuffed bits are removed if identified.

2.2. Literature Review of Bit Stuffing Techniques

The bit stuffing nature of CAN protocol make the transmission time a complex function and varies length of the message. It is very hard to predict the message transmission time and to implement in a real time embedded system. To address this problem T. Nolte and Nahas proposed XOR masking technique. They proposed a simple XORing scheme with a bit masking of alternate zeors and ones (101010). Original data was XORed with the bit mask before transmission and at the receiver side the received data was XORed with the same mask for reconstruction of the original data but the entire CAN frame format was XORed with the key value dynamically changed the original message ID of the messages [4] [5]. Hence making it very difficult to implement on a real-time system as the source and designation addresses changed during receiving.

Another research work Inversion bit stuffing mechanism was proposed by Maha in which, the proposed system helped in reduction of stuffed bits in the data filed only. Bytes by byte the data's were checked and if the potential bit stuffing was identified in a byte fifty bit was made complimentary. One byte in the data filed was reserved to carry flag to identify the byte location of inversion taken place. In this work the message ID integrity was maintained but at a cost of reducing message payload, as the key of encoding has to be carried along with the original message ID, made the proposal unrealizable for real time CAN data traffic [6].

Imran Sheik and M. Short research work proposed methods to reduce the impact of bit stuffing like XOR, the transmitting data and at receiver side the data is being XOR once again to reconstruct the original information and by over clocking method also research has been made to improve speed of CAN data transmitted [7] [8]. All the alternative methods proposed are not back ward compatible and the feasibility of implementation has also been reduced because of additional hardware requirement. To overcome the above stated limitations and to effectively utilize the available CAN bandwidth, a novel method of short Controller area network has been proposed in this research work. Research work 1 demonstrates the selective XORing technique to reduce the bit stuffing impact on the protocol. Research work 2 mathematically proved the sort CAN proposal that reduces the maximum payload defined in the CAN protocol from 8 byte to 4 byte in which, the worst case response time of the CAN network can be reduced. Research work 3 shows how these techniques could be implemented on a virtual network simulator and real time hardware.

3. CAN Real Time Implementation

3.1. In Vehicle Architecture

Even in a low cost Light Commercial Vehicle, LCVs, minimum of 10 to 15 ECUs (Electronic Control Units) are interconnected to each other using different serial communication protocols like CAN (Controller Area Network), LIN (Local Interconnect Network), Flex ray and etc. Every automobile has its own electrical architecture and the design would be owned by the OEMs.

The vehicle architecture shown in **Figure 1** illustrates how the instrument cluster is interfaced with Engine ECU, GPS Systems and with the Body control module which acts as a CAN gateway between other sensor units and actuators. CAN gateway could interface LIN and MOST as per the requirement. All powertrain related ECUs are connected with CAN1 and hence it is known as power train CAN and all infotainment related CAN is connected in CAN2 called infotainment CAN.

3.2. CAN Format

Bosch published several versions of the CAN specification and the latest is CAN 2.0 which is published in 1991. This specification has two parts; part A is for the standard formats with an 11-bit identifier, commonly called CAN 2.0A and part B is for the extended format with a 29-bit identifier, called CAN 2.0B [9].

The CAN frame that has been extended and standard frame format are shown in **Figure 2** and the CAN versions are listed in the **Table 1**. SOF stands for Start of Frame which indicates the beginning of Data Frames and Remote Frames consisting of a single dominant bit. Arbitration is different for standard as well as for extended frame format. In standard and extended frame format, RTR bit stands for Remote Transmission Request for DATA frame. RTR is 0 where as for REMOTE frame RTR is 1. RTR helps in identifying the type of the frame.

In extension frame format 11 bit base ID is followed by SRR bit. SRR stands for Substitute Remote Request bit and it is a recessive bit that helps in resolving the collusion of Standard and extension frame formats. IDE bit

Table 1. CAN versions versus factures.

Version	Features
CAN 2.0A	• Sends and Receives Only Standard (11 Bit) Messages • "Trashes" Extended (29 Bit) Messages on the Bus (Sends Error Frames)
CAN 2.0A/2.0B Passive	• Sends and Receives Only Standard (11 Bit) Messages • Will not "Trash" Extended (29 Bit) Messages on the Bus
CAN 2.0B	• Sends and Receives Both Standard and Extended Messages • Can Have a Mix of Both Message Types

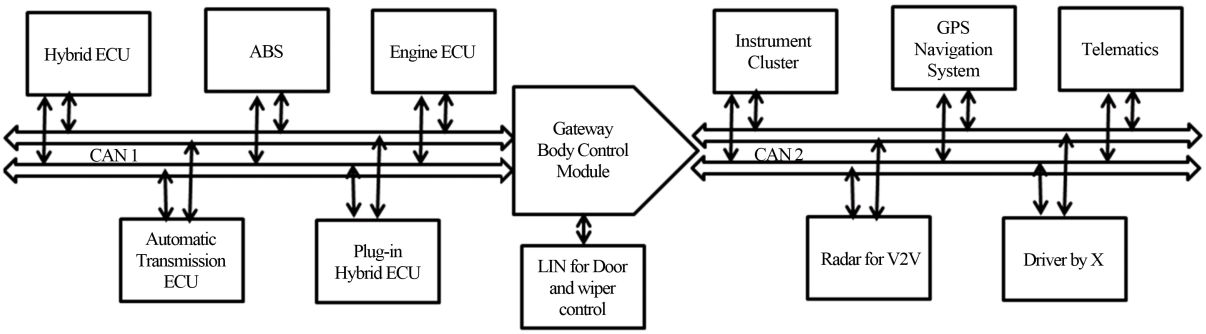


Figure 1. In vehicle network architecture.

Bus Idle	SOF	Arbitration Field	Control Field	Data Field	CRC SEQUENCE	CRC DELIMITER	ACK Field	EOF	Inter Mission
	1 Bit	12/32 Bit	6 Bit	0 to 8 Byte	15 Bit	1 Bit	2 Bit	7 Bit	3 Bit

Standard Frame Format					
SOF	IDENTIFIER 11 BITS	RTR	IDE	R0	DLC

Extended Frame Format								
SOF	IDENTIFIER 11 BITS	SRR	IDE	IDENTIFIER EXTENSION 18 BITS	RTR	R1	Ro	DLC

CONTROL FIELD for Standard Format and Extended Format					
IDE / r1	r0	DLC3	DLC2	DLC1	DLC0

Figure 2. CAN frame format.

is followed by SRR/RTR which stands for Identifier Extension Bit. This bit actually indicates whether the frame format is standard or an extension version. The IDE bit transmitted in the Standard Format is "dominant", whereas in the Extended Format the IDE bit is recessive. Control bit consists of six bits in which IDE is used to identify the standard and extended frame formats. R0 and R1 are reserve bits. These reserved bits have to be sent as "dominant", but receivers accept "dominant" and "recessive" bits in all combinations. DLC stands for Data Length Code which is 4 bit wide used to represent the data length 0000, stands for 0 bytes and 1000 stands for 8 byte.

Depending upon the DLC length, the data filed would carry the required data which each contain 8 bits which are transferred MSB first. **Table 2** lists the different types of fames in CAN communication.

4. Bit Stuffing

In CAN message transmission, bit stuffing is done at the transmission side and removed at the receiver end. During bit stuffing a non information bit is inserted by the controller during the data sequence of 5 similar bits which would be a complimentary logic to the sequence as shown in **Figure 3** Bit stuffing is done to limit the

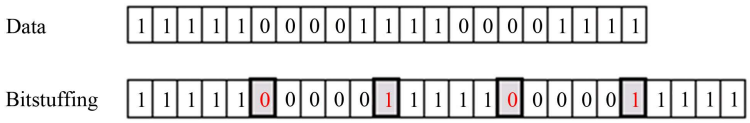


Figure 3. CAN bit stuffing.

Table 2. CAN frame types.

S. NO	Frame Types	Function
1	**A Data Frame**	Carries data from a transmitter to the receivers.
2	**A Remote Frame**	Transmitted by a bus unit to request the transmission of the Data Frame with the same Identifier.
3	**An Error Frame**	Transmitted by any unit on detecting a bus error.
4	**An Overload Frame**	Used to provide for an extra delay between the preceding and the succeeding Data or Remote Frames.

number of consecutive bits of the same value in the data to be transmitted and the receiver does not need any extra information about the location of the bit stuffing in order to do the de-stuffing. Any violation of this rule is understood as a bit stuffing error in the network.

4.1. Impact of Bit Stuffing

In a given CAN standard frame format, the worst case bit size is 135 bit and the maximum possible bit stuffing could happen during the transmission is 24 bits, similarly in extended frame format the worst case bit size is 160 bit and 29 possible bit stuffing could happen. If the CAN network is operating at 1 Mbps, a single CAN bit timing is 1 micro second. Totally due to bit stuffing character of CAN network the possibility of worst case response time has been increased to 29 micro seconds. **Table 3** lists CAN bit stuffing impact in standard and extended frame format.

4.2. XORing Technique

Imaran Shiek and M. Short made enhancement of, the XOR technique for bit stuffing reduction [7]. The valuable finding of their work has motivated us to implement the same technique to the standard CAN SAEJ1939. But unfortunately the XORing techniques proposed by Imaran and Short is applied to the entire CAN frame which has modified the CAN message ID dynamically. The SAE J1939 standard does not allow message ID to be modified. The technique has been inherited by applying the XORing technique only to the pay load of 64 bit. A significant improvement has been found in the net bit error rate and made the proposed system compatible to the automotive standard.

The dynamic CAN message frame is subjected to XOR masking during transmission and decoded with the same key at the receiver. **Figure 4** explains the XORing technique which is applied during the transmission for the entire frame and similarly decoded at the receiver end. This technique does not require any key to decode at the receiver end.

4.3. Selective Bit Stuffing Algorithm

In Selective bit stuffing, the only the data filed is exposed for the bit stuffing, Two bit mask are used BM1: [55 55 55 55 55 55 55 55] Hex and BM2: [91 91 91 91 91 91 91 91] Hex. The Algorithm compares the transmission data with bit masking value if found similar the data are transmitted without masking and at the receiver side the data are compared with masking vale. Whenever the data received is found similar to masking the data's, the received data is taken without XORing the mask. if the data is different from the masking value the masking logic is being executed as per the flow chart.

In selective XORing method the bitmask is applied for the data field in the selective CAN frames. The bit mask values used are BM1: [55 55 55 55 55 55 55 55] and BM2: [91 91 91 91 91 91 91 91]. The bitmask is applied for the CAN frames other than the worst case scenarios as mentioned above. If the data is the same as

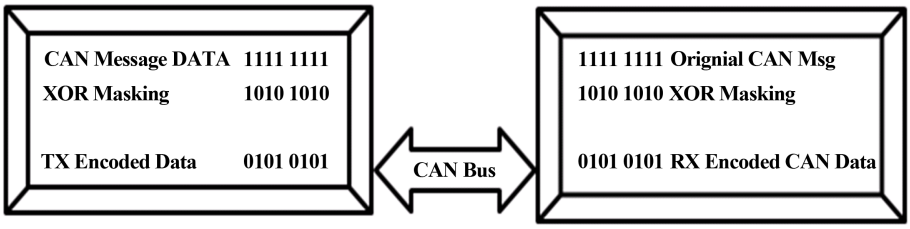


Figure 4. CAN XORing technique.

Table 3. CAN bit stuffing.

Type	Message ID size in bits	Total bit size in bits	Maximum possible bit stuffing
CAN 2.0A (No of bits)	11	135	24
CAN 2.0B (No of bits)	29	160	29

the bitmask or the complement of the bitmask it will be transmitted without masking. The algorithm explains this method.

4.4. CAN Analysis

Tindell [10] proposed methods to calculate worst case latencies of CAN frames. Naturally the analysis is based on fixed priority response time analysis. The calculations are focused on worst case queuing patterns of frames. Each message in a CAN network is assigned with a fixed priority. In order to analyze the worst case behavior of each message is streamed and to compute the network load, which has S set of messages stream $S_m \in S < P_m$, T_m, $C_m >$ where P_m is priorly defined in the message identifier, T_m is the time period of the message and C_m is the worst case transmission of message m. the worst case latency R_m of a CAN frame that is sent on stream S_m with assumption of minimum variation in queuing time to be zero is defined by

$$R_m = J_m + q_m + C_m \tag{1}$$

The calculation of the worst case response time of each message in the CAN network is guided by Response time analysis and hence it could be compared with the dead line of the message to verify the schedule of messages. Three different parameters that influence worst case response time are queuing jitter J_m, queuing delay q_m and transmission time C_m.

Queuing jitter J_m—longest time between event initiation and message being queued before ready for transmit on the CAN bus.

Queuing delay q_m—maximum time that a message spends in the CAN node before successful transmission on the CAN bus.

Transmission time C_m—maximum time that a message could take for transmission.

Inherited from sender's task, the effective queuing time is given by

$$q_m^n = B_m + \sum_{j \in hp(m)} \left[\frac{q_m^{n-1} + J_j + T_{bit}}{T_j} \right] \times \left(C_j + 3 \times T_{bit} \right) \tag{2}$$

where

C_j is the transmission time of message j

T_{bit} is the bit time

$hp(m)$ is the high priority set of messages than that of m

$lp(m)$ is the low priority set of messages than that of m

$3 \times T_{bit}$ is considered as inter frame gap as a port of the data frame represented in Tindell calculation.

Maximum blocking time or worst case blocking time of a frame sent on S_m

$$B_m = \max_{k \in lp(m)} \left(C_k \right) + 3 \times T_{bit} \tag{3}$$

The maximum blocking time of a CAN message happens when a low priority message starts to transmit just before a message m is placed when it is ready to transmit. This message m has to wait for the low priority message to transmit completely and bus goes to idle. The time duration is represented as B_m called maximum blocking time.

4.5. CAN Utilization Factor Analysis

PengHao and Yang Shun [2] proposed the computation method of hard real time scheduling utilization rate indicated that if requested capacity of CAN is not above 30% then time deadlines of all messages flow are met

The utilization of the message time is defined as

$$U = \sum_{i=1}^{m} \frac{C_i}{T_i} \tag{4}$$

The CAN standard specifies that the maximum data payload is 8 byte and lest is at 0 byte. S_m denotes the maximum number of bytes of payload of a packet and T_{bit} specifies the bit time of a single *bit*.

Transmission time of a CAN message is represented as

$$C_m = \left\{ g + 8 \times S_m + 13 + \left[\frac{g + 8 \times S_m - 1}{4} \right] T_{bit} \right\} \tag{5}$$

For CAN 2.0A has g value of 34 and CAN 2.0B has g as 54, where g is the number of frame header subjected to bit stuffing. Maximum transmission time is denoted as $C_{\max}$ and minimum transmission time is denoted as C min. Applying $S_m = 0$ and $S_m = 8$ on equation (5) we get

$$C_{\max} = \left\{ g + 64 + 13 + \left[\frac{g + 63}{4} \right] T_{bit} \right\} \tag{6}$$

$$C_{\min} = \left\{ (g + 13) T_{bit} \right\} \tag{7}$$

To understand the impact of payload versus the CAN unitization, the payload size has been applied on above equation 6 and 7 to compute the utilization factor. **Table 4** has been computed to understand the payload impact which clearly indicates that there is a significant improvement in unitization rate of standard and extended frame formats. The utilization rate of CAN for standard is 26% and extended is 29%, it is understood that all the messages in the network are schedulable under this value and without any deadline missed.

5. Short CAN-Payload Reduction Technique and Analysis

The next research work focus on Payload reduction and implementation strategies for real time requirement. CAN payload vary from 0 bytes to 8 bytes (maximum). In a bandwidth constrained CAN network, this work

Table 4. CAN utilization rate versus payload.

Bytes	C min Std	C min Ext	C Max Std	C Max Ext	Utilization Rate Std	Utilization Rate Ext
0	47	67	55.25	80.25	45.9	45.5
1	47	67	65.25	90.25	41.2	42.6
2	47	67	75.25	100.25	38.4	40
3	47	67	85.25	110.25	35.5	37.7
4	47	67	95.25	120.25	33	35.7
5	47	67	105.25	130.25	30.8	33.9
6	47	67	115.25	140.25	28.9	32.3
7	47	67	125.25	150.25	27.2	30.8
8	47	67	135.25	160.25	25.7	29.4

focus on optimum payload size to handle bandwidth requirement. This proposed short works in payload size from 0 byte to 4 byte maximum. Payload reduction in any networked communication increases the network utilization as the overhead for a single packet is being reduced. The above study about the controller area network protocol follows in the same fashion. Optimum payload for the real time communication protocol like CAN has to be of four bytes. Because both standard and extended protocols have almost same utilization factor of 33 and 35.7, planning larger number will have the same impact on the utilization factor.

In CAN standard and extended protocol version it has been defined that the Message pay load size various from 0 to 8 bytes and in short CAN it is defined to be 0 to 4 bytes. **Table 5** shows clearly that whenever the payload size of the CAN is reduced from 64 to 32, the maximum transmission time of CAN 2.0A reduces from 135 to 95 and for extended CAN frame format CAN 2.0B, the transmission time has been reduced from 160 to 120. The calculation is made for 1 Mbps gross bit rate and one bit is computed as 1 µSec.

In the proposed Short CAN frame format, all the Start of Frames (SOF) and Message Identifiers which arbitrate with other messages follow the same standard protocol of 11 bits for standard and 29 bits for extended CAN frame format. Control bit is 6 bits, CRC 15 and CRC delimiter, ACK delimiter and ACK slot shares one bit each. Maximum bit stuffing values are taken for the computation purpose.

5.1. Short CAN over SAEJ1939 Benchmarking

In order to benchmark and implement the findings of the research work carried out, SAE J1939 Protocol controller application layer is enhanced to short CAN model. SAE (Society of Automotive Engineers) J1939 is a set of standard one that is used in heavy duty commercial vehicles like buses and Truck. The physical layer is described in the J1939/11 explained about the electrical interface [11]. The data link layer is described in the J1030/21 explaining about the message construction, bus access, arbitration and error transmission. The application layer is described in the J1939/71 and about the data content in individual message is explained in J1939/73. These are pre defined messages and location of information in each bit of the CAN message has been clearly spelt out and implemented by automobile OEMs.

J1939 protocol has been implemented over CAN2.0B protocol which has 29 bit identifier for priority and source address and designation address assignment. The message ID consists of "PDU" called as Protocol Data Unit and PDU1 stands for destination address and PDU2 stands for broadcast message. Every bit in the SAE J1939 Message has a meaning and function, **Table 6** indicated every bit location and its function.

Table 5. Short CAN frame format and maximum transmission time.

Bit Name	CAN 2.0A	CAN 2.0B	Short CAN 2.0A	Short CAN 2.0B
SOF	1	1	1	1
Identification bits	11	11	11	11
SRR	x	1	x	1
IDE	x	1	x	1
Identification bits	x	18	x	18
RTR	1	1	1	1
Control bit (Max)	6	6	6	6
Data bits	64	64	32	32
CRC	15	15	15	15
CRC delimiter	1	1	1	1
Bit stuff max	24	29	16	21
ACK slot	1	1	1	1
Ack delimiter bit	1	1	1	1
End of frame	7	7	7	7
Interframe space	3	3	3	3
Total (Maximum time µSec)	135	160	95	120

Table 6. SAE J1939 message ID and its fuctions.

Message ID	Bit	Function
Priority	3 bits	Defines message priority during arbitration
Reserved	1 bit	This bit is reserved for future, always it must be set as zero
Data page	1 bit	expansion of possible parameter group
PDU format (PF)	8 bits	PF indicate message transmitted with a destination address or always broadcast message
PDU specific (PS)	8 bits	If PF is between 0 and 239 then PS contains addressable MSG PDU1 If PF is between 240 and 255 then PS contains only broadcast MSG PDU2
Source Address	8 bits	Source Address
SAE J1939		**Total 29 Bits**

5.2. SAEJ1939 Message Classification

SAEJ1939 Protocol has a well defined application layer in which there are 27 inseparable messages whose lengths of the messages may fall in between 4th and 5th byte of CAN payload. The detailed survey of SAEJ1939 protocol clearly indicates that there are 27 such messages that have to be handled with care while converting into short CAN Message.

In order to categorize the short CAN message implementation under the Heavy commercial vertical standard protocol SAE J1939, we have take 256 messages (PGN) described in the standard. **Figure 5** shows the combination of messages that are available in the automobile standard. To test the effectiveness of the short CAN we have selected 27 messages that will be configured in Matlab Simulink model and test the functionality and its peak load impact along with the short CAN messages. **Table 7** highlights the 27 inseparable messages under the standard. The PGN number of these messages and it corresponding length of the message has been listed for direct transmission of 8 bytes as they are not modifiable in to short CAN because of its data location in the payload given in the standard. The other 229 SAEJ1939 messages are converted in to short CAN and testing was performed over it.

6. Experimental Setup

Model based system design and development is a fast developing powerful design and analysis tool. The proposed instrument cluster design is developed using vehicle network toolbox which provides real-time connectivity to the Matlab Simulink models, CAN transceivers and vector virtual CAN bus driver.

Model based designing is the latest design methodology in the field of automobile electronics. As the embedded system software development depends on the High level languages like C, Embedded C, assembly level languages and etc, development any small logic or control is completely depending upon the programming capability of a particular software development team. To overcome effective software development with limited programming knowledge, model based software development has opened the doors of aspirant hardware engineers to program the desired functionally in a common development platform like Matlab where actual control or logic could be designed using Simulink blocks and the developed algorithm could be implemented in the target platforms

6.1. Implementation of Short CAN

SAEJ1939 sends data in CAN 2.0B in 29 bit extended message ID format and the location of sensor and other vehicle related data are defined as per the specification of the application layer. If the data contains only 2 bytes of information other 6 bytes of information are filled with all ones causing potential bit stuffing location. These types of messages could be reduced to short CAN of 4 byte information and control bit could be modified as short CAN message and other message be truncated. **Figure 6** shows the short convertor model helps us to identify potential short CAN message and truncate the rest of the unwanted bits.

There are totally 246 predefined SAEJ1939 Messages in which 27 are more than 4 byte of single messages. If these messages are splitted in two real time data cannot be reconstructed. So these 27 messages are transmitted in CAN network without any modification. Rest of the message could be split into two or if the useful information is

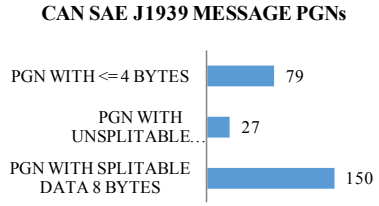


Figure 5. CAN SAE J1939 messages PGN types.

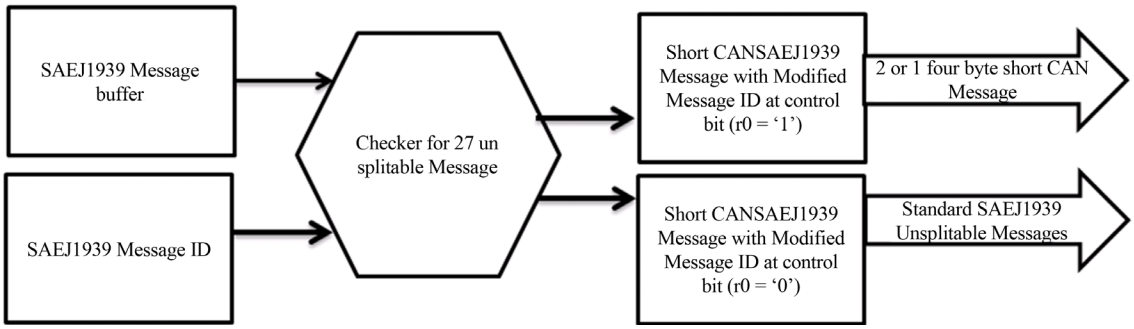


Figure 6. Short CAN block diagram.

Table 7. SAE J1939 inseparable messages.

S. No	NAME	PGN	Length	Parameter Name
1	Anti-theft Status	PGN 56320	2 bits	Anti-theft Encryption Seed Present Indicator
2	Anti-theft Request	PGN 56576	2 bits	Anti-theft Encryption Indicator States
3	Electronic Engine Controller 1	PGN 61444	4 bits	Engine Torque Mode
4	Vehicle Dynamic Stability Control 2	PGN 61449	2 bytes	Steering Wheel Angle
5	After Treatment Historical Information #1	PGN 64920	4 bytes	After Treatment 1 Total Fuel Used
6	After Treatment Historical Information #2	PGN 64921	4 bytes	After Treatment 2 Total Fuel Used
7	Farebox Service Detail	PGN 64956	2 bits	Farebox Service Status
8	Transit Milepost	PGN 64959	1 byte	Number of bytes in the Milepost Identification
9	ECU Identification Information	PGN 64965	Variable-up	ECU Part Number
10	Operators External Light Controls Message	PGN 64972	4 bits	Work Light Switch
11	FMS-standard Interface Identity/Capabilities	PGN 64977	2 bits	FMS-standard Diagnostics Supported
12	Engine Torque History	PGN 65168	1 byte	Number of Engine Torque History Records
13	Trip Time Information 2	PGN 65200	4 bytes	Trip Cruise Time
14	Trip Time Information 1	PGN 65204	4 bytes	Trip Time in VSL
15	Engine Speed/Load Factor Information	PGN 65207	2 bytes	Trip Maximum Engine Speed
16	Trip Fuel Information (Gaseous)	PGN 65208	4 bytes	Trip Drive Fuel Used (Gaseous)
17	Trip Fuel Information (Liquid)	PGN 65209	4 bytes	Trip Drive Fuel Used
18	Trip Distance Information	PGN 65210	4 bytes	Trip Distance on VSL
19	Trip Fan Information	PGN 65211	4 bytes	Trip Fan On Time
20	Compression/Service Brake Information	PGN 65212	4 bytes	Total Compression Brake Distance
21	Software Identification	PGN 65242	1 byte	Number of Software Identification Fields
22	Retarder Configuration	PGN 65249	4 bits	Retarder Type
23	Engine Configuration	PGN 65251	2 bytes	Engine Speed At Idle, Point 1 (Engine Configuration)
24	Vehicle Weight	PGN 65258	8 bits	Axle Location
25	Component Identification	PGN 65259	5 bytes	Make
26	Power Takeoff Information	PGN 65264	1 byte	Power Takeoff Oil Temperature
27	Ambient Conditions	PGN 65269	1 byte	Barometric Pressure

only 4 byte length, rest of the unwanted messages are truncated.

6.2. Short CAN Instrument Cluster Model

Matlab target Simulink model with the power of vehicle network tool box provides the programming power to the hardware designers. Vehicle Network Toolbox provides connectivity to CAN devices from MATLAB and Simulink. The tool box has the capability of generating required CAN Messages in the required format from a CAN Node and receive the CAN message in other node.

Tools have additional features like decoding; encoding, filtering, logging and recording of CAN Message, It could also be connected with CAN dbc files and replay the sequence. Vehicle network tool box supports Vector, Kvaser, and National Instruments interface hardware and Vector CAN database (.dbc) files which can be recorded and play back. CAN communication could be configured using Matlab command line or from Simulink model. **Figure 7** shows the Matlab simulation model which simulates the virtual vehicle instrument cluster and follows the short CAN over SAE J1939 standard and the results shows no performance degradation because of payload reduction of 8 byte to 4 bytes. The Virtual CAN enabled instrument cluster model simulates 2 virtual CAN nodes in data collection is done with Simulink car engine model designed by Simulink and transmit by virtual CAN node 1. The CAN node 2 receive the simulated data of vehicle performance during riding is given to a CAN node 2 enabled Simulink GUI. Vehicle parameters like odometer, trip meter, Engine RPM, Fuel level and tell tale (LED indicator) are modeled using upgraded short CAN model. These were no performance degrading between SAE J1939 messages and modified SAE J1939 Messages over short CAN [12].

6.3. Short CAN Workbench Setup

As the performance of the short CAN was tested using virtual model. The same short CAN messages were coded in the CAN enabled multifunction display. The multifunction display was powered by automobile graded freescale controller. The experimental setup shown in **Figure 8** had CAN debugger and CANoe CAN bus simulator. The 256 short CAN SAE J1939 messages were modeled and performances were verified.

Similarly the second research work on selective bit stuffing was also tested on the workbench setup. The CANoe, AHU, MFD and SYNC are connected to a common bus. The selective bit stuffing method is analyzed for the various worst case scenarios. Interestingly the Selective bit stuffing technique gave constant 8 bit stuffing for the applied combination. The selective bit stuffing algorithm has reduced the maximum bit stuffing impact in

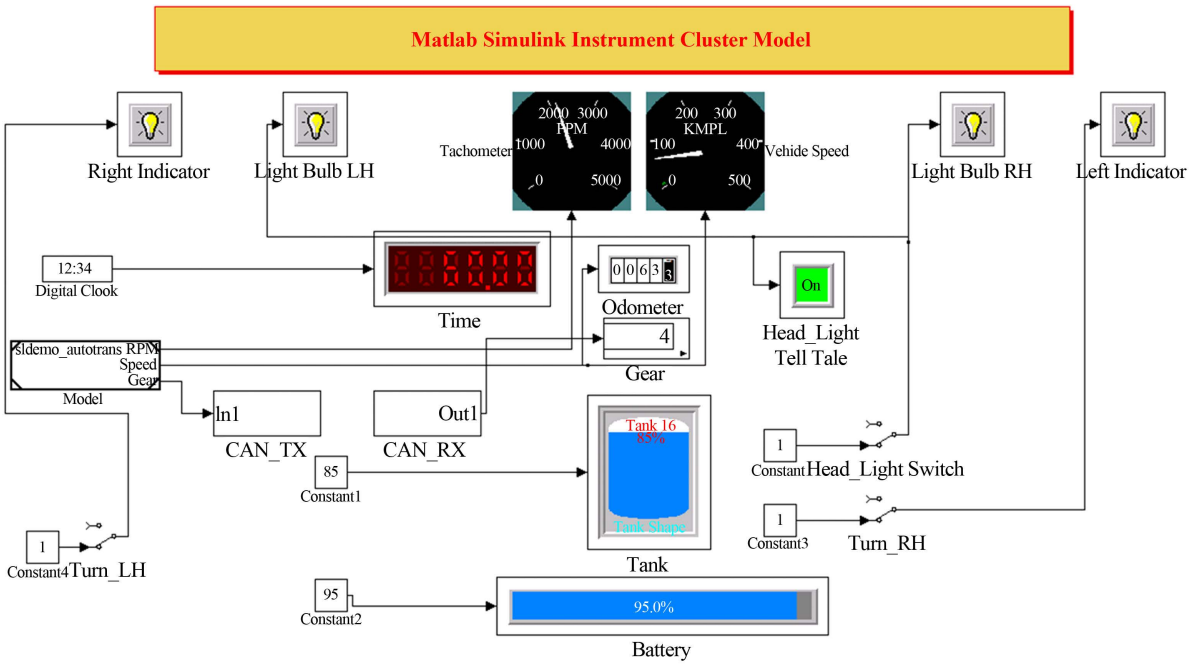


Figure 7. Virtual instrument cluster model.

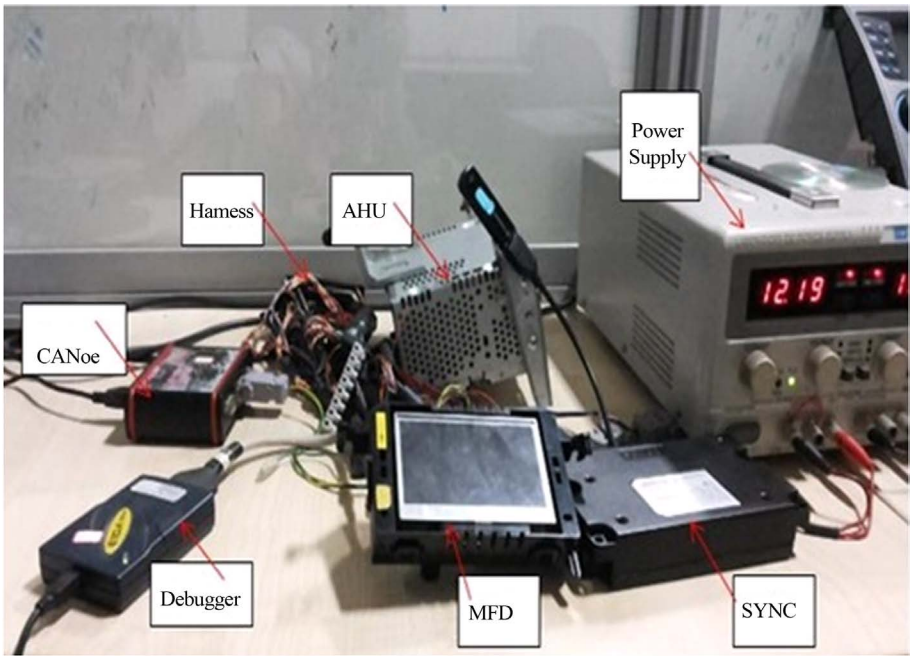


Figure 8. Workbench setup for performance evaluation.

payload from 24 bits to 8 bits in standard CAN messaging scheme and from 29 to 13 in extended frame format.

7. Results and Discussion

7.1. Peak Payload

For analysis, 50 messages out of 256 listed CAN SAE J1939 standard were configured and tested in the test setup, as only 50 CAN SAE messages were allowed to be configured from a single node. The Peak load was captured in the CAN tool and different combinations of CAN messages were transmitted and the peak load is observed over CAN tool. 5 non splittable messages are sent and the transmission is executed for 1 time step in the Matlab. Because of payload size reduction we were able to reduce the peak load in a real time CAN traffic. **Table 8** shows the impact on peak load during testing.

7.2. Bit Stuffing Reduction

Impact of bitstuffing over payload can be easily understood from the bit stuffing versus payload analysis. Total number of bits after bit stuffing for the worst case scenario using the selective XORing method is (114 + 8 = 122).

The largest frame takes 119 bit times to send the data using this method. This is reduced to 122 bit times from 138 bit times for CAN 2.0A. For CAN 2.0B the largest frame takes 144 bit times to send the data. This is reduced to 144 bit times from 160 bit times. The possible number of stuff bits of each byte for CAN 2.0A and CAN 2.0B is shown in the following **Figure 9**. The short CAN proposal combined with selective XORing will have a greater impact in the worst case response time of CAN messaging.

7.3. Selective Bit Stuffing Net Bit Rate

Net bit rate is defined as the number of useful bits carried per second. The CAN communication protocol carries various overheads like Message ID, CRC, bit stuffing and control bits. The impact of net bit rate could be reduced by applying selective bit stuffing technique. **Table 9** discusses the computation of net bit rate.

The experimental results have clearly indicated that the maximum possible bit stuffing using selective bit stuffing is 8. The experiments were conducted for all b4 byte combination. The proposed algorithm was stable and was not introducing any additional bit which helped in improving the net bit rate of CAN messages. The

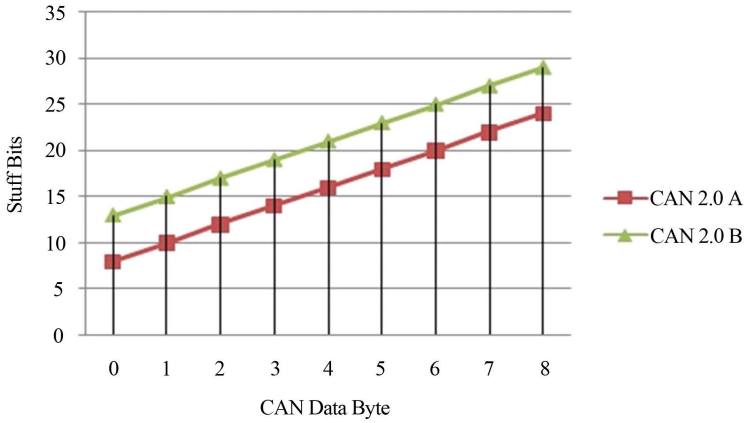


Figure 9. Impact of bit stuffing.

Table 8. Real time CAN data analysis results for peack load.

S.NO	SAE J1939 Messages	Payload	Peak load (%)
1	All messages with extended frame format and message	8	**90.56**
2	Message no 6 to 20 message	4	**81.77**
3	Message no 11 to 20 messages are splitted (message ID + 1)	4	**89.12**

Table 9. Net bit rate calculation with selective bit stuffing.

CAN 2.0A Maximum Length of the Frame for the Worst Case Scenario after Selective Bit Stuffing	CAN 2.0B Maximum Length of the Frame for the Worst Case Scenario after Selective Bit Stuffing
Frame length = 114 + (max no of stuff bits) = 114 + 8 (with selective bit stuffing technique) = 122	Frame length = 131 + (max no of stuff bits) = 131 + 13 (with selective bit stuffing technique) = 144
Net bit rate = ((11 + 1 + 4 + 64)/122) * 1 Mbps = (80/122) * 1 Mbps = 655.73 kbps	Net bit rate = ((29 + 1 + 4 + 64)/144) * 1 Mbps = (98/144) * 1 Mbps = 680.55 kbps

same analysis was performed with different CAN message payloads. The selective bit stuffing was having approximate 650 kbps for standard and 680 kbps for extended CAN frame format.

Selective bit stuffing algorithm results for various payload has been listed in **Figure 10**. The Net bit rate has increased from 579 kbps to 655.73 kbps for standard CAN 2.0A and for 680.55 kbps for extended CAN 2.0B frame formats.

7.4. Short CAN Utilization Factor

The proposed Short CAN is a shorter version of native CAN and could be implemented on the existing CAN network provides back ward compatible that exists in the same commercial available network. Results shown in **Figure 11** clearly demonstrates short CAN of 4 byte increase CAN bus utilization factor by achieves 6% increase extended format and 8% in standard CAN frame format of 8 bytes.

8. Conclusion

This paper presents a novel short CAN technique for effective utilization of available CAN bandwidth and explores the impact of selective bit stuffing over data payload without altering major modification in the existing CAN protocol. Various experiments have been conducted on different payload sizes using Matlab Simulink and the influence of payload; bit stuffing, worst case response time and busload have been studied. Smaller the payload size is, smaller the bus load is. There was no performance degradation in Simulink working model simulation and real time CANoe hardware with multi functional display has been shown on the study on 266 PGN

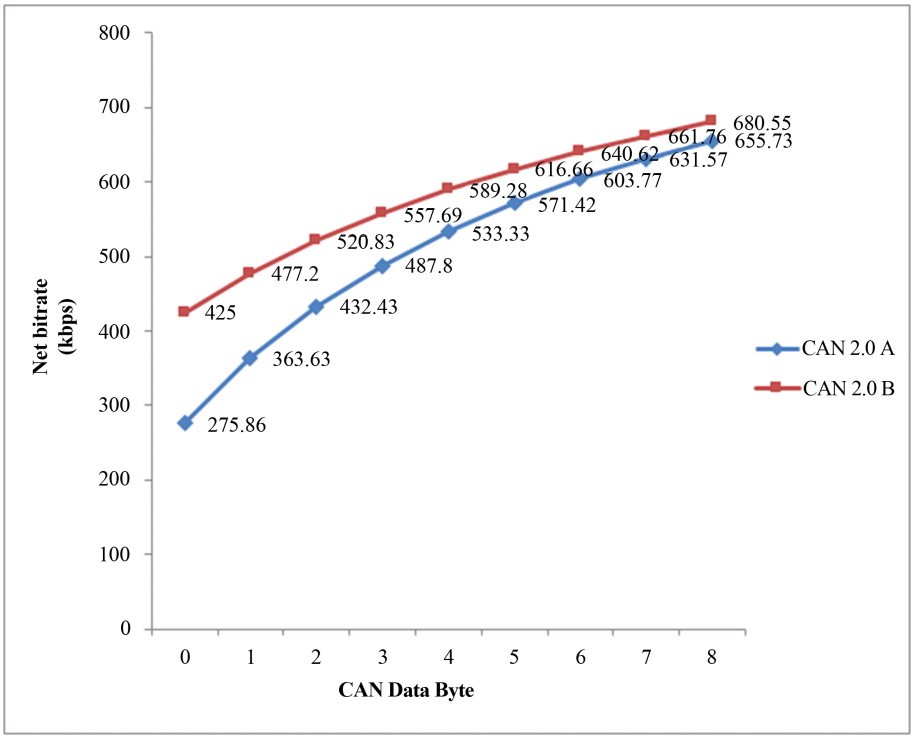
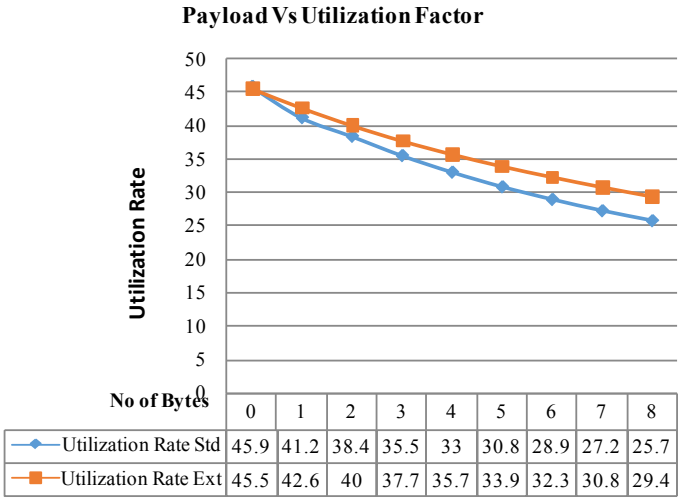


Figure 10. CAN data payload impact on net bit rate.



No of Bytes	0	1	2	3	4	5	6	7	8
Utilization Rate Std	45.9	41.2	38.4	35.5	33	30.8	28.9	27.2	25.7
Utilization Rate Ext	45.5	42.6	40	37.7	35.7	33.9	32.3	30.8	29.4

Figure 11. CAN utilization factor.

messages. The implementation of short CAN is possible without any additional hardware and resources as clearly indicated by benchmarking with SAEJ1939, thereby improving the protocol as backward compatible.

Future Work

The future work of the research work is to integrate short CAN technique and selective bit stuffing technique on a single VLSI CAN IP core for ASIC and FPGA prototyping.

References

[1] Najafzadeh, S., Ithnin, N., Razak, S.A. and Karimi, R. (2014) BSM: Broadcasting of Safety Messages in Vehicular Ad

Hoc Networks. *Arabian Journal for Science and Engineering*, **29**, 777-782. http://dx.doi.org/10.1007/s13369-013-0686-y

[2] Hao, P. and Shun, Y. (2010) Computing Utilization Rate of Rate Monotonic Scheduling Mechanism in Controller Area Network. 2010 *IEEE International Conference on Vehicular Electronics and Safety* (*ICVES*), Qingdao, 15-17 July 2010, 133-137. http://dx.doi.org/10.1109/ICVES.2010.5550934

[3] Steve C. Talbot and ShangpingRen (2009) Comparison of Field Bus Systems, CAN, TTCAN, Flex Ray and LIN in Passenger Vehicles. 29*th IEEE International Conference on Distributed Computing Systems Workshops*, *ICDCS Workshops*'09, Montreal, 22-26 June 2009, 26-31. http://dx.doi.org/10.1109/ICDCSW.2009.15

[4] Nahas, M., Pont, M. and Short, M. (2009) Reducing Message Length Variations in Resource Constrained Embedded Systems Implemented Using CAN Protocol. *Journal of Systems Architecture*, **55**, 344-354. http://dx.doi.org/10.1016/j.sysarc.2009.03.004

[5] Nolte, T., Hansson, H., Norstrom, C. and Punnekkat, S. (2001) Using Bit Stuffing Distribution in CAN Analysis. *IEEE/IEE Real-Time Embedded Systems Workshop*, London.

[6] Hassan, M.M. (2015) Bit Stuffing Techniques Analysis and a Novel Bit Stuffing Algorithm for Controller Area Network. *International Journal of Computer Systems*, **2**, 80-87.

[7] Sheik, I. and Short, M. (2009) Improving Information Throughput in Controller Area Networks: Implementing the Dual Speed Approach. *Proceedings of* 8*th International Workshop on Real Time Networks*, Dublin, June 2009, 57-62.

[8] Sheikh, I. (2011). Doctor Thesis of Philosophy, University of Leicester, Leicester.

[9] Bosch, R. CAN Specification 2.0. Technical Report, Robert Bosch-Gmbh.

[10] Tindell, K., Burns, A. and Wellings, A.J. (1995) Calculating Controller Area Network (CAN) Message Response Times. *Control Engineering Practice*, **3**, 1163-1169. http://dx.doi.org/10.1016/0967-0661(95)00112-8

[11] Vinodh Kumar, B. and Ramesh, J. (2014) Automotive in Vehicle Network Protocols. 2014 *International Conference on Computer Communication and Informatics*, Coimbatore, 3-5 January 2014, 1-5.

[12] Kokkolaras, M. (2004) Simulation-Based Optimal Design of Heavy Trucks by Model-Based Decomposition. *International Journal of Heavy Vehicle Systems*, **11**.