American Journal of Embedded Systems and Applications
2014; 2(4): 29-37
Published online August 20, 2014 (http://www.sciencepublishinggroup.com/j/ajesa)
doi: 10.11648/j.ajesa.20140204.11

Design and evaluation of controller area network for automotive applications

Abu Asaduzzaman, Sandip Bhowmick, Md Moniruzzaman

EECS Department, Wichita State University, Wichita, Kansas, USA

Email address:

Abu.Asaduzzaman@wichita.edu (A. Asaduzzaman), sxbhowmick@wichita.edu (S. Bhowmick), mxmoniruzzaman@wichita.edu (M. Moniruzzaman)

To cite this article:

Abu Asaduzzaman, Sandip Bhowmick, Md Moniruzzaman. Design and Evaluation of Controller Area Network for Automotive Applications. *American Journal of Embedded Systems and Applications.* Vol. 2, No. 4, 2014, pp. 29-37.
doi: 10.11648/j.ajesa.20140204.11

Abstract: Now-a-days, as computerization of automotive increases, the need for more/advanced electronic control units (ECUs) is growing. Vehicles are increasingly behaving like computers with wheels. Due to the abundance use of ECUs in cars, electronic system design is becoming complicated day by day. As a consequence, information processing in such complex systems faces interruptions. Therefore, there is a need for a system which will be the connective tissue between a vehicle's computers and different sensors. Controller Area Network (CAN) is such an embedded system to bring communication and connectivity in automobile system together. Optimization of CAN bus is a concerning issue because of the recent demands such as hybrid, electric propulsion, or driver assistance that involves more stringent real-time constraints. In this paper, we introduce an optimized CAN bus for vehicle automation through microcontroller based design and implementation. Multiple modules of a vehicle are modeled using CAN bus. The proposed CAN bus system is evaluated by analyzing the frame response time using "RTaW-Sim:" a CAN simulation and configuration tool. Experimental results show that the proposed CAN system helps decrease the frame response time by up to 45% when compared with a traditional system. The CAN bus simulation platform using RTaW-Sim also helps reduce the design cost and improve the systems quality.

Keywords: CAN Module, Controller Area Network, Frame Response, Microcontrollers, RTaW-Sim

1. Introduction

The automobile industry has faced great development in the 21st century. Vehicle system is made up of automotive electric architectures consist of a large number of ECUs carrying out a selection of managed functions [1, 2]. Developing electricity preserving and low emission merchandises becomes both directions of automobile industry. In automobile system, we generally expect less fuel consumption, greater safety, more comfort, and better pollution management. As a response, automakers started to roll out innovative attributes. In addition, new technology with improved computing capacity is available. However, the competition among the automakers is growing day-to-day. Therefore, automakers are adopting the most recent technology to make their vehicles get noticed in a busy marketplace [2]. Recognizing the demand for increased operations in automobiles, researchers adopt intelligent computing systems, such as embedded systems, in cars. An embedded system is a microprocessor system with a number of committed functions, generally with real-time computation constraints.

A modern vehicle may have many ECUs in different subsystems for various purposes. Different such subsystems include air bags, anti-lock braking, engine-control, audio systems, windows, doorways, mirror alteration, etc. Many of these subsystems are independent or reliant subsystems. Improved electronic control system can improve relaxation and vehicle dynamics. However, there are some new challenges such as the body wiring sophistication, space constraints, and reliability dilemmas. Communicating among dependent subsystems is also crucial. In order to address these issues, the car network engineering concept has been developed [3-6]. In automobile networking protocols, some prerequisites must be satisfied, which contain important reduction of wiring harness, lowering body fat and prices, bettering the effectiveness of low latency times, fault diagnosis and settings flexibility, and accentuating the level

of intelligent control [7, 8]. Subsystems necessitate the exchange of particular performance and position advice within defined communication latency. CAN bus can offer a simple but effective real-time communication protocol between controls, actuators, sensors, and other subsystems. In this work, a CAN bus protocol is used with a microcontroller system for car automation.

2. Literature Survey

Since 1970s, there is an exponential increase in the variety of electronic systems that have replaced those which are only mechanical or hydraulic. Reliability of hardware parts and the improving performance due to software technologies enable implementations of complicated functions. As a result, the safety and the comfort of the vehicle's occupier are enhanced. Particularly, one of the primary motives of electronic systems is to assist the driver to manage the vehicle through capabilities related to the steering system, grip (i.e., manage the driving torsion), or braking such as antilock braking system (ABS), electronic equilibrium software (EES), electrical power steering (EPS) system, lively suspensions, and engine-control. Another reason for using electronic systems is to control apparatus within the body of a vehicle; for example, lights, wipers, doors, windows, amusement, and communicating gear (e.g., radio, DVD, hands free phones, and satnav systems). At the beginning of automotive electronics, each new function is executed as a stand-alone ECU, which is a subsystem composed of a microcontroller and a set of detectors and actuators. This tactic quickly proved to be insufficient with the need for functions to be dole out over several ECUs and the importance of information exchanges capabilities. In today's luxury cars, up to 2500 pieces of information are exchanged by up to 70 ECUs. Until the beginning of the 1990s, information was changed through point-to-stage links between ECUs. These problems are addressed by using networks where the communicators are multiplexed over a shared medium, which therefore demand defining rules and protocols for handling communications, particularly, for granting bus accessibility. Today, it has become the most broadly used network in automotive systems and it's estimated that the quantity of CAN nodes offered per year is now around 400-million [1].

2.1. Automobile OnBoard Diagnostics

From early 1980s, cars have already been designed with engineering that assists to determine issues related to digital engine systems. This technology, called OnBoard diagnostics (or OBD), refers to a vehicle's self-diagnostic and reporting capability. OBD is composed of numerous devices that exchange their results to some special nodes in the shape of analytical trouble codes within the car's computer system. A fundamental OBD system consists of an ECU (short for electronic or engine control unit) as depicted in Figure 1. ECU utilizes input signal from various sensors to manage the actuators (e.g., gas injectors) to get the desirable functionality. The “Check Engine” light, also known as the Malfunction Gauge Light (MIL), supplies an early warning of malfunctions to the car owner. A modern vehicle can support numerous parameters, which can be generated via the Diagnostic Link Connecter (DLC) using an apparatus called a scan-tool. (MIL and DLC are not shown in the figure.) While the automobile will be powered on, the OBD-II program screens various output signals and motor states. When the OBD-II program finds a trouble within the discharge control program, a dash light is lit suggesting “test the engine”. An affiliated diagnostic trouble code is saved in the PC’s storage to record the exceptional issues and states of the discharged control parts. The diagnostic problem signal tips could be retrieved by the computer using a check device. Through the uses of these records, an accurate tech-repair is possible; other troubles may also be uncovered precisely.

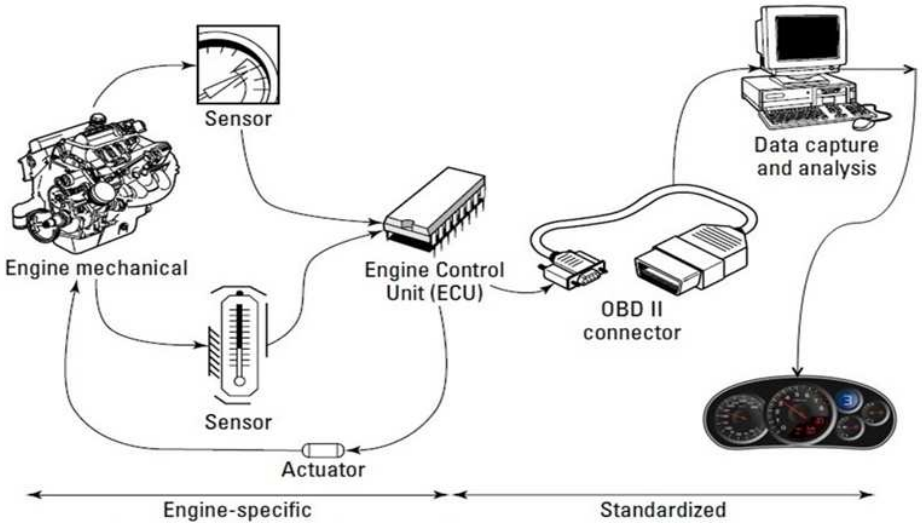

Figure 1. *Schematic diagram of the OnBoard diagnostic.*

2.2. OBD-II with New Technology

The CAN is the newest communication program within the automotive planet. CAN is a way of linking each of the digital systems within a vehicle together allowing them to communicate with each other. As OnBoard computers improve, so does the variety of distinct electronic systems. The information processed and recorded by each management module is regularly used by one or more management modules on the program. A necessary standardized means of swiftly passing information between the modules is required to improve the CAN bus performance. The CAN bus can do a lot in automobile communications. In general, CAN bus systems have the potential to truly make changes in the process and can conduct communication between parts faster. The unique OBD-II/EOBD protocols (J1850 PWM, J1850VPW, ISO-9141 and ISO-14230) help reduce the variety of tools needed for engine and/or emissions fault diagnosis. However, by permitting four protocols there was none the less a great deal of confusion and over-complexity of tools. Simplifying this to an individual protocol can merely lead to lessen the cost of diagnostic tools. So multiplexing technologies and specifically CAN, rise up very fast, and help keep the wiring harness complexity under control while satisfying the growing demand for data broadcasting. The use of CAN bus can enhance the OBD-II system by replacing the following criteria:

- Reduced wiring requirements and fast communication between data bus.
- Increased electrical reliability due to the reduction of electrical connectors and reduced use of redundant sensors.
- It helps to master the complexity of the architectures.
- It reduces the hardware cost, weight, consumption, space, lower vehicle weight, etc.
- It facilitates an incremental design process.
- It leads to better communication performance and helps to match the bandwidth needs. Sometimes, a 60% loaded CAN network can be more efficient than two 40% CAN networks interconnected by a gateway causing delays and high jitters.

3. Controller Area Network

The Controller Area Network (i.e., CAN) is a serial communication protocol for networking detectors, actuators, and other nodes in real-time systems. In this section, we discuss the basic description of CAN including network layer, rule of bus arbitration, and its error handling mechanisms. Extensions of time oriented greater-level protocols and CAN, including time-triggered CAN are explained in Figure 2. CAN communication protocol data is passed between different parts of CAN module on a network and conformed by the Open Systems Interconnection (OSI) model. The physical layer of OSI model defines the actual communication between devices connected by the physical medium. The ISO 11898 architecture (as shown in Figure 2) defines the lowest two layers of the seven layer OSI/ISO model as the data-link layer and physical layer. These two layers are mostly used for CAN bus communication [2].

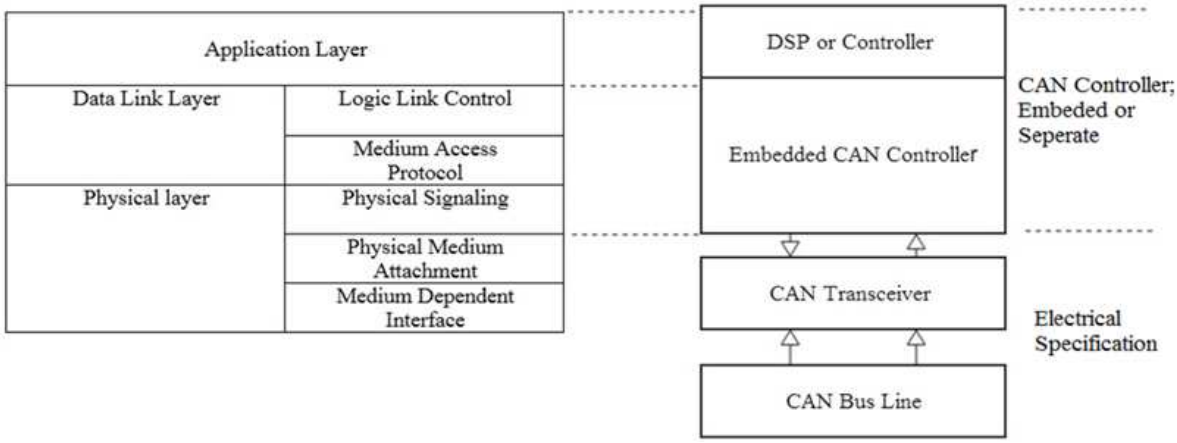

Figure 2. *Application layers of ISO Standard 11898 architecture.*

3.1. CAN Bus Message Formats

CAN distinguishes four message formats: data, distant, malfunction, and overload frames. Here, we restrict the discussion into the information frame. An information frame starts with the start-of-frame (SOF) bit. The identifier and the remote transmission request (RTR) bit form the arbitration subject. The management field consists of six bits and signifies how many bytes of information follow in the information field. The information field is accompanied by the cyclic redundancy checksum (CRC) discipline, which empowers the receiver to verify if the received bit sequence was corrupted. The 2-bit acknowledgment (ACK) area is used by the transmitter for an acknowledgment of a valid frame from any receiver. The ending of a frame is signaled through a seven-bit end-of-frame (EOF). In addition, there is an extended data frame with a twenty nine bit identifier.

3.2. CAN Bus Arbitration

Arbitration is the mechanism that handles bus access conflicts. Whenever the CAN bus is free, any unit can start to transmit a message. Possible conflicts, due to greater than one unit beginning to transmit concurrently, are resolved by bit shrewd arbitration utilizing the identifier of each device. During the arbitration phase, each transmitting device transmits its identifier and compares it with the amount monitored on the bus. The system continues to transmit, if these amounts are equivalent.

3.3. CAN Bus Error Handling

Error detection and error management are significant for the operation of CAN. Error detection is done in five distinct manners in CAN: (i) Bit monitoring, (ii) bit stuffing, (iii) frame check, (iv) ACK check, and (v) CRC. Bit tracking means each transmitter monitors bus level; the bus level malfunctions if the level doesn't agree with the transmitted signal. After having transmitted five indistinguishable bits, a node will consistently transmit the reverse bit. This additional bit is ignored by the receiver. During the ACK in the concept frame, all receivers are presumed to send a dominant bus. If the transmitter, which conducts a recessive level, doesn't detect the dominant level, then an error is indicated by the ACK check mechanism. In comparison to other network protocols, this mechanism leads to high data integrity and a short error recovery time. CAN bus, therefore, supplies procedures for malfunction handling, including re-transmission, and re-initialization.

3.4. Scheduling of CAN Bus

CAN protocol implements fixed priority scheduling of CAN messages. Higher priority node has lower node identification (ID). If available bandwidth is insufficient, then the problem occurred is traditional fixed-priority established scheduling. It is possible that the low precedence control iterations cannot access the network on a regular basis, since the limited resources have been used by high priority loops. As an effect of incredible delays, low precedence control iterations may be destabilized.

The trouble of given priority may be defeated by the use of immediate feedback scheduling algorithm, namely maximum urgency first (MUF). MUF is integrated in the community scheduler. Upon invocation, the scheduler computes the urgency of each manage iteration centered on current system output signals and the set points. According to the MUF algorithm, new precedence is produced by the scheduler according to the urgency values. Then, messages in distinct iterations will be aired in accordance with the newly assigned precedence [3]. A fresh mixed traffic schedule (MTS) is centered on the communicating principle of controller area network, community scheduling, and investigation of program. The heart thought of MTS is to place the comparative deadline tips into the identifier [4]. The earliest deadline first (EDF) concept scheduling algorithm is used for high precedence tips and the rate monotonic scheduling (RMS) concept scheduling algorithm is used for low precedence tips.

3.5. Reliability of CAN Bus

Reliability refers no failures in an operating time. High malfunction management capacity of CAN enhances system dependability. Transmission is aborted by the node forcibly, if a malfunction has been found by any concept transmitting node. Afterward, it tries to re-transmit repeatedly till its concept is transmitted efficiently. To avert such catastrophe, the transmit malfunction counter (TEC) and the receive malfunction counter (REC) are began to diagnose the states of CAN control. CAN control has REC and TEC which improves dependability of CAN bus system [5]. A CAN controller can be in one of three states: error active, error passive, or bus off state. Once the CAN controller has entered bus off state, it must be reset by the host microcontroller in order to be able to continue operation.

4. Controller Area Network for Vehicle Automation

Controller area network (i.e., CAN) provides high dependability and good real-time performance with very low cost. It is a vehicle bus standard designed to allow microcontrollers and devices to communicate with each other within a vehicle without a host computer [6]. The robustness and performance of the CAN technology, as well as the new possibilities brought by distributed software functions, have motivated engineers to use more and more bandwidth in order to improve existing Electrical and Electronic (EE) functions and introduce new ones. Data communication in automobile regularly has many sensors transmitting modest information packets.

4.1. Prototype Design

The CAN is a multi-master broadcast serial bus standard for connecting ECUs. A prototype design, as shown in Figure 3, illustrates how electronic components in distributed control architecture for cars are connected by CAN. Some of these form independent subsystems, but communications among others are essential.

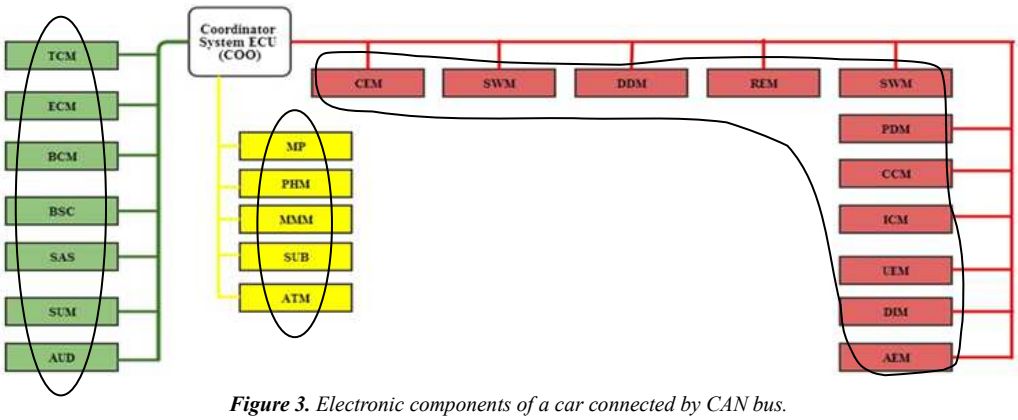

Figure 3. *Electronic components of a car connected by CAN bus.*

Table 1. *ECU Parameters*

Powertrain and Chassis	TCM: Transmission Control Module ECM: Engine Control Module BCM: Brake Control Module BSC: Body Sensor Cluster SAS: Steering Angle Sensor SUM: Suspension Module
Infotainment/Telematics	MP1,2: Media Players 1 and 2 PHM: Phone Module MMM: Multimedia Module SUB: Subwoofer ATM: Antenna Tuner Module
Body Electronics	CEM: Central Electronic Module SWM: Steering Wheel Module DDM: Driver Door Module PDM: Passenger Door Module REM: Rear Electronic Module CCM: Climate Control Module ICM: Infotainment Control UEM: Upper electronic Module DIM: Driver Information Module AEM: Auxiliary Electronic

The blocks represent ECUs and the thick lines represent networks. There are three classes of ECUs: power-train and chassis; infotainment; and body electronics. The ECUs are defined by their acronyms. Several networks are used to connect the ECUs and the subsystems. There are three different CAN bus subsystems. CAN bus subsystems are connected via a coordinator (COD) system. The maximum configuration for the vehicle contains about 40 ECUs [6]. Some important ECU parameters are briefly discussed in Table 1.

4.2. Block Diagram for CAN Bus Communication

The block diagram for CAN bus communication developed in this study is shown in Figure 4. Node monitors on bus and information is send to computer for display and further investigation. CAN analyzer is used for fault investigation of the program also as evaluation. The parts which are connected by a CAN community are normally sensors, actuators, and other control apparatus. These devices are not connected right to the bus, but through a host chip and a CAN controller. CAN nodes include host chip, CAN controller, and CAN transceiver. An individual node is linked to can-bus line. Following are the main component of a CAN bus:

Host Processor: The host processor decides which messages it wants to broadcast itself and what is the significance of received messages. It manages devices and sensors. Actuators can be linked to the host chip.

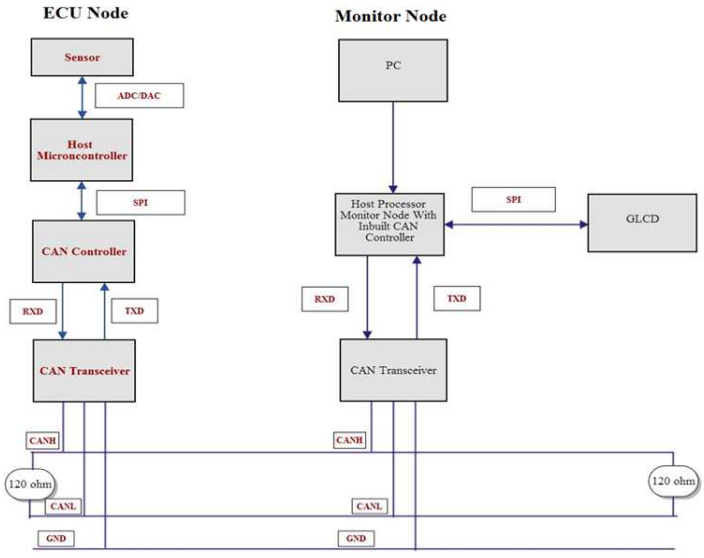

Figure 4. *Block diagram for CAN communication.*

CAN Controller: The CAN control stores received bits serially in the bus until a complete message can be acquired, which may subsequently be brought by the host CPU. The host CPU stores the transmit messages.

Transceiver: It accommodates signal level in the bus to degree that the CAN control has protecting circuitry that shields the CAN control and anticipates. It converts the transmit-bit sign received in the CAN control into a signal which is sent onto the bus [7]. CAN transceiver is an interface between the real bus and protocol control.

4.3. CAN Bus Software Protocol Design

In the proposed CAN bus prototype for vehicle automation, we consider a host microcontroller and a CAN controller. The software component of principal program unit is broken up into method initialization device, CAN initialization component, message sending unit, message receiving unit, and interrupt support unit. System initialization unit comprises initialization of host processor and computer screen node. Work-flow diagram for the CAN node is illustrated in Figure 5.

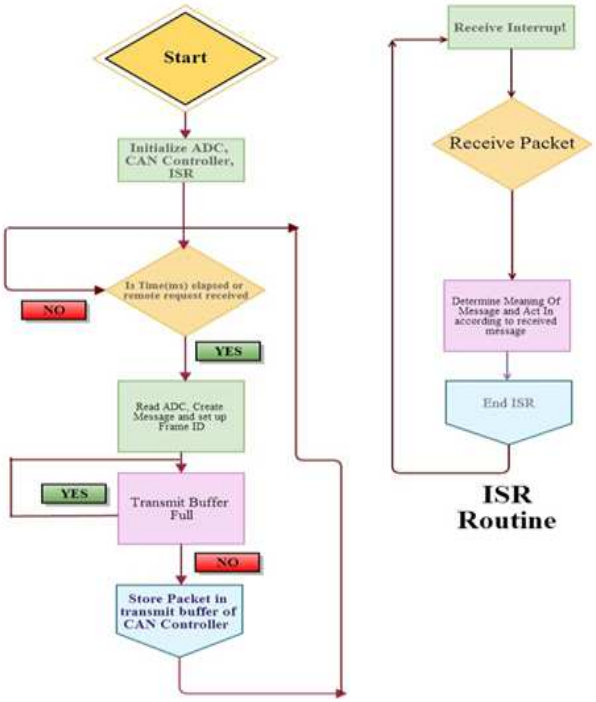

Figure 5. *Work-flow of the proposed CAN node.*

CAN Controller Initialization: CAN control initialization procedure includes initialization of internal register of CAN controller, such as initialization of transmit and obtain buffer. When initializing the CAN registers in the CAN controller, the system configures the controlling way, bit time filter, mask register, and compose buffer; immediately clears the study; and interrupts enable register.

CAN Message Sending: After initialization, the CAN controller is prepared to work and is in standard communicating status. The CAN concept sending embraces inquest way or it sends concept when distant request is obtained. In inquest mode, packet is created by host processor and sends information to CAN Control transmit buffer for additional transmission.

CAN Message Receiving: There are two means for messages getting – interrupt way and inquest way. In the system, the interrupt mode is executed. In interrupt mode, if CAN controller receives legitimate message in obtain buffer, it creates interrupt signal on INT pin. The message is prepared by the host processor by utilizing SPI instructions of CAN controller [7].

5. Simulation Criteria of CAN in Automotive Design

In this section, we present a simulation technique for frame response that happens around the network nodes at runtime. As CAN bus could be the principal communication method between Electric Control Units (ECUs) in vehicles, CAN frame response times are now being thoroughly researched. Frame response time distributions were eventually provided richer information on the real-time behavior of the system. So having the ability to get correct frame response time distributions may result in substantial cost optimization on hardware.

5.1. Assessing Frame Response Time Distribution

Frame response times fundamentally rely on the frame characteristics and of the phasing involving the interacting nodes. Upper bounds on the response times, regardless the phasing between ECUs, can be computed with analysis tools such as RTaW-Sim. Reply times while in the standard event may be better assessed by simulation. The conventional simulation method will be to gain data on the variety of adjustments equivalent to unique original designs. This really is required just because a simple simulation work merely conveys a small pair of response times for each frame, while the period of the machine is small (usually a couple of seconds) [9]. As the phasing between the ECUs varies continually because of the clock drifts, simulating just one trajectory is enough to see an extensive range of response situations for every single frame. Because there is an individual simulation run to do, this method is easier from an experimental point of view.

5.2. Simulator Toolset: Real Time at Work (RTaW-Sim)

RTaW-Sim is a timing accurate simulator of CAN networks that enables the designer to come up quickly with the best design and configuration choices. RTaW-Sim is also useful to assess the performance and reliability of a CAN bus system. In our experiments, we use RTaW-Sim Starter edition 1.4.4. RTaWSim provides a comprehensive and precise performance evaluation of CAN networks, and enables to investigate scenarios and compare the impact of different design and configuration alternatives. It is powered

with rich graphical edition and visualization environment with communication architecture editor, distribution plots, and Gantt diagrams.

6. Simulation

In this section, we present step by step simulation processes and frame response times related to the expected behavior of the network [10]. This is the most elementary and most beneficial characteristic of the CAN simulator since it provides an exact idea of the frame response times for a CAN bus. The experimental results depict the expected performance improvement.

It should be remarked that the microcontroller clocks may have drifts, as it happens, then it is obvious to understand the length of time. Figure 6, a screen-shot, shows the CAN connections between the electronic components of a car. In the experiment, the speed of the bus in kbits/s and is explained by the tool-tip of "Speed".

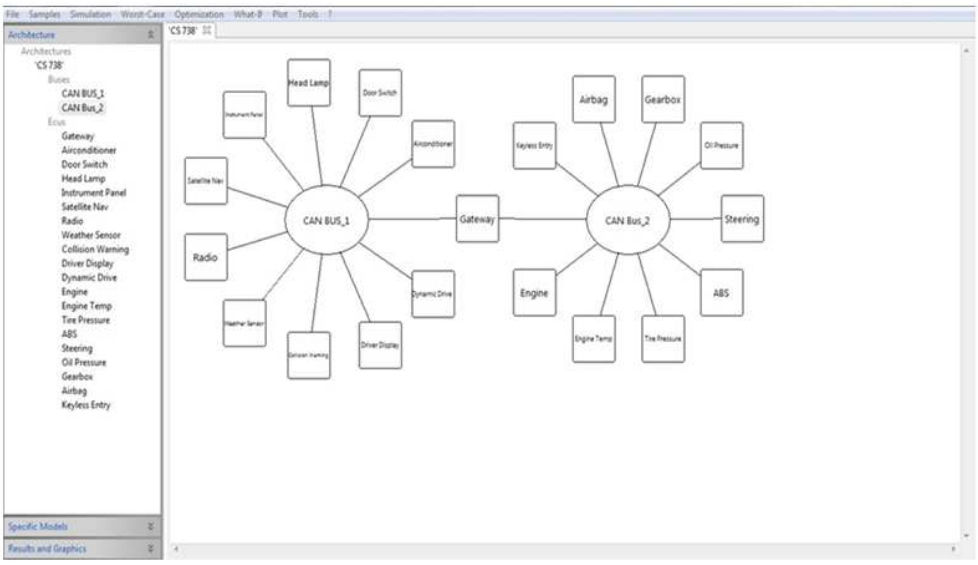

Figure 6. *Components of a car as connected by CAN buses through a gateway.*

The periodic bus load is displayed for the chosen bit-stuffing: "10%" means that the bit-stuffed part of the frame is 10% longer than nominal, with the worst-case option. As shown in Figure 7, we select "Length & Sample Times" and enter "Intermediate Statistics" times, where snapshots of the (evolving) response time statistics will be taken: 30 sec, 10 min, 1 hour, and 1 day. When the simulation ends, a second dialog pops-up and gives information about the completed simulation. The statistics are stored both in the RTaW-Sim input file and in csv files in the indicated folder. The obtained statistics can be visualized in the CAN bus tab, by choosing the corresponding bus simulation configuration "BusSimConfig". A sample time statistics is shown in Figure 8.

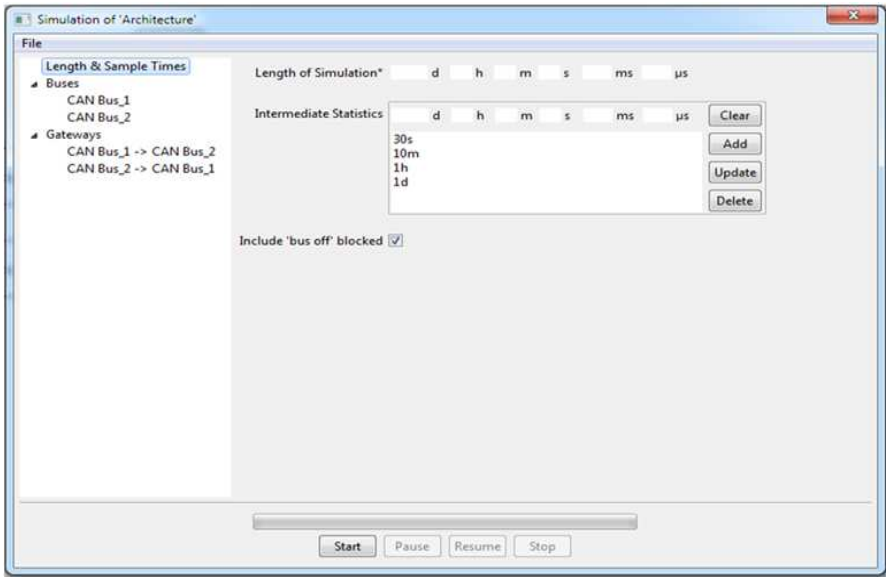

Figure 7. *Intermediate Statistics time selection.*

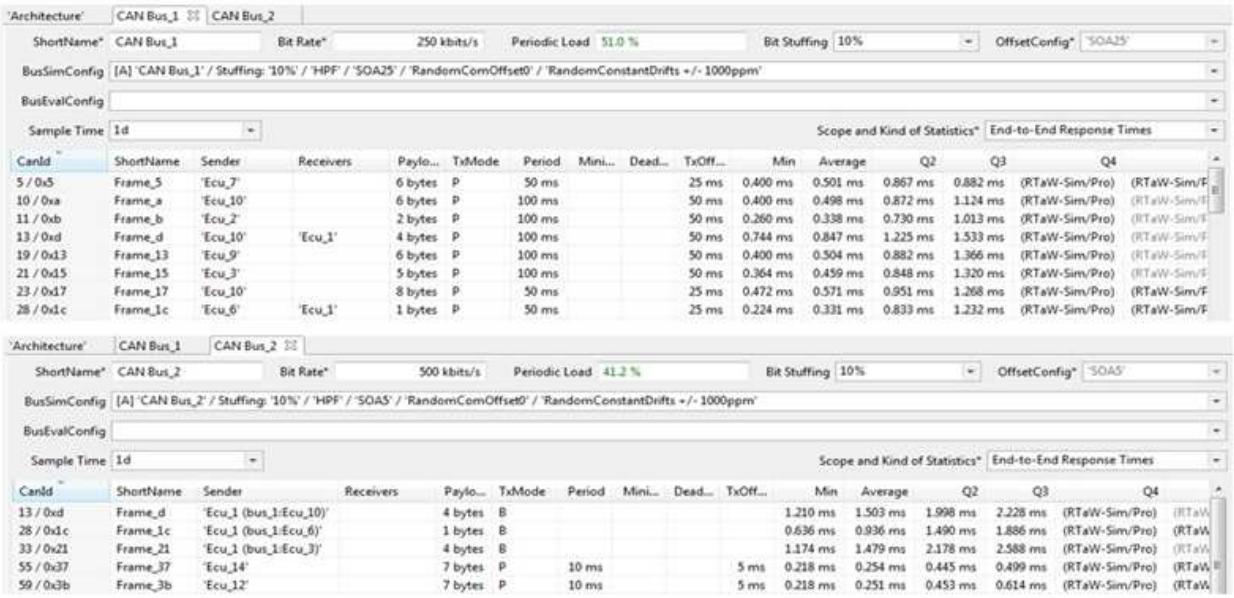

Figure 8. *Simulation results: time statistics.*

Frame response times are illustrated in Figures 9(a) and 9(b) for CAN bus 1 and bus 2, respectively. Here, x-axis represents the number of frames and the y-axis represents their response times in milliseconds. The five graphs/lines in Figure 9(a) correspond to the statistics: Minimum, Average, Maximum, and two quantiles. The two quantiles are $(1\text{-}10^{-2})$ quantile and $(1\text{-}10^{-3})$ quantile. A $(1\text{-}10^{-n})$ quantile is a threshold such that the probability that a response-time of the frame is larger than that threshold, is lower than 10^{-n}. For the frames where only the minimum is visible, the average and maximum have the same value as the minimum, meaning that the response time of the frame is always the same – of course this is not the typical behavior and it may be due to the fact that station clock drifts, variable bit-stuffing for the frames, and transmission jitters are not modeled in this study.

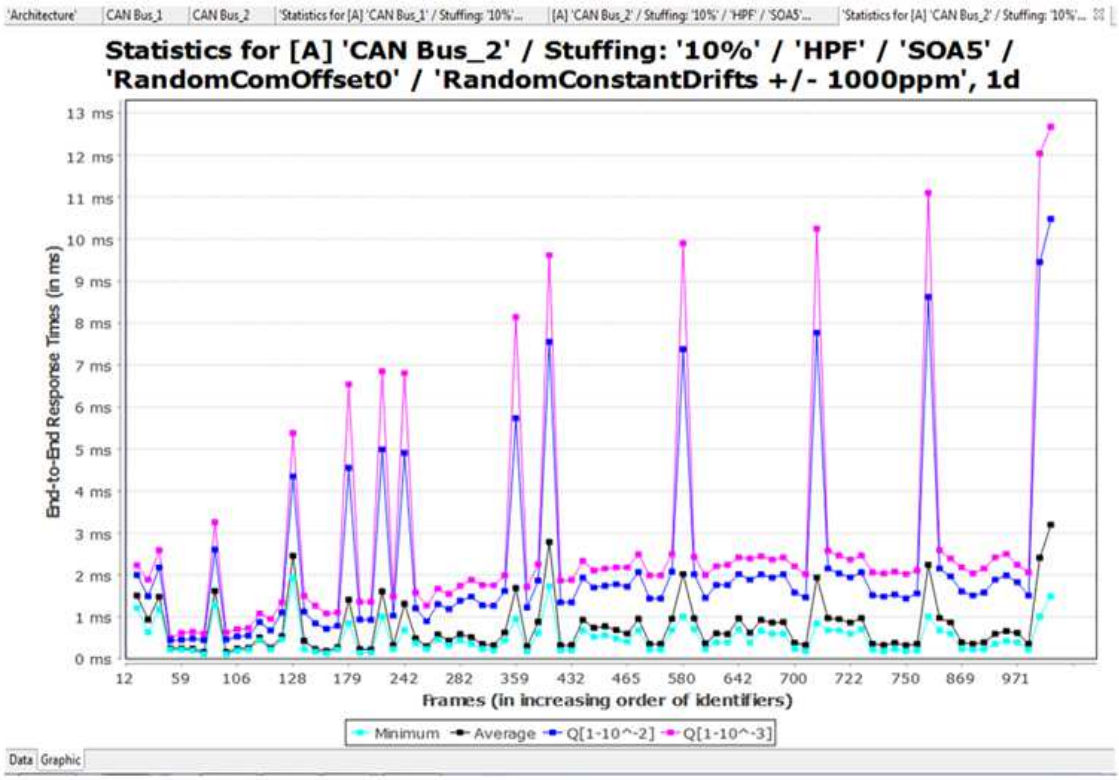

Figure 9(a). *Statistics of CAN bus 1.*

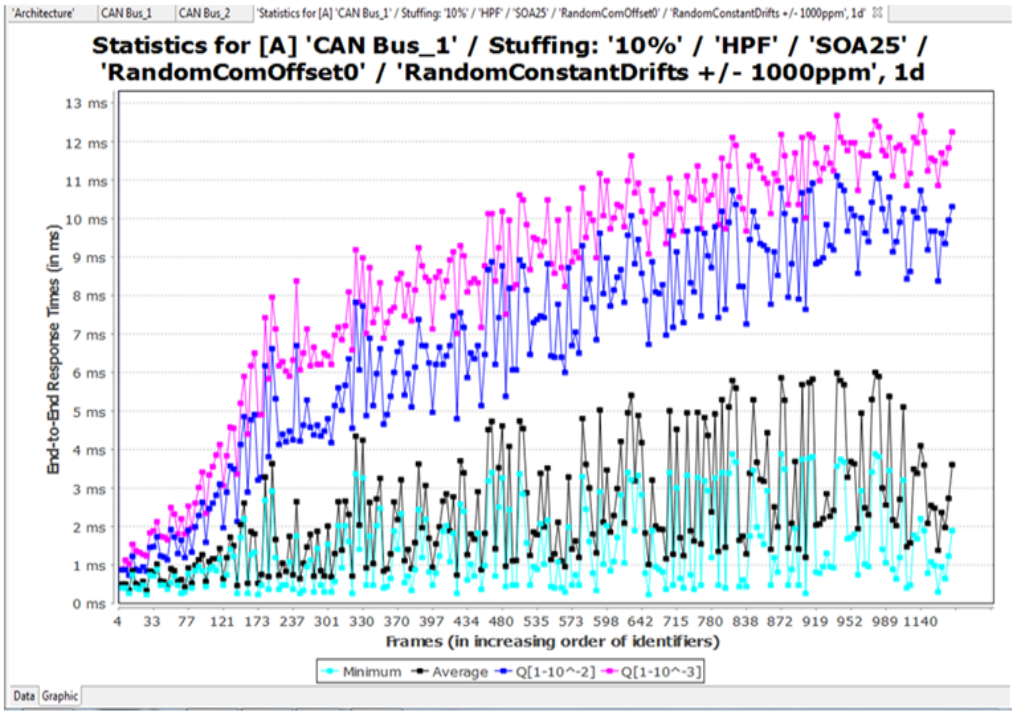

Figure 9(b). *Statistics of CAN bus 2.*

From Figures 9(a) and 9(b), the most observed response times (the top two lines) may differ depending on the initial configuration. This is especially true for the lowest priority frames. As expected, the minimum response times (the most bottom line) are almost equal because the phasing between sending nodes changes in such a way that every frame will be sent without delay. The fruitful outcome is that the average response times (the second line from the bottom) are almost equal. This result is meaningful because it suggests that for a long simulation time, the initial phasing is not significant for the final response time distributions.

7. Conclusion

This paper provides a brief discussion about CAN bus implementation in automobiles. Real-time, reliability, and flexibility – all these characteristics make CAN bus an icon network communication technology applied in automobile network communication field. In this work, CAN-bus based communication system for vehicle automation is simulated. Experimental results show that the CAN bus has potential to save time and design cost. We plan to investigate a CAN bus system with variable clocks which are available in the Pro-Version of the simulation tool (RTaW-Sim).

References

[1] N. Navet, S. YeQiong, F. Simonot-Lion, C. Wilwert, "Trends in Automotive Communication Systems," IEEE proceedings, Vol. 93, No. 6, pp. 1204-1223, 2005.

[2] K. Johansson, M. TÃrngren, et al, "Vehicle applications of controller area network," Handbook of Networked and Embedded Control Syst., 2005.

[3] F. Xia, X. Dai, Z. Wang, and Y. Sun, "Feedback Based Network Scheduling of Networked Control Systems," IEEE proceedings, 2005.

[4] S. Fan, J. Du, H. Sun, and T. Liang, "Research on Mixed Traffic Scheduling of Networked Control Systems Based on CAN Bus," IEEE DOI 10.1109/ICINIS.2009.64, 2009.

[5] C.E. Lin and H.M. Yen, "Reliability and Stability Survey on Can-Based Avionics Network for Small Aircraft," IEEE DOI 7803-9307-4/05/2005, 2005.

[6] J. Fr¨oberg, K. Sandstr¨om, et al, "A comparative case study of distributed network architectures for different automotive applications," Handbook on Information Technology in Industrial Automation, CRC Press, 2004.

[7] A.S. Shinde and V.B. Dharmadhikari, "Controller Area Network for Vehicle Automation," Int'l J. of Emerging Tech. and Advanced Engineering, 2(2), 2012.

[8] G.S. Kumar, "Designing and Development of a CAN Bus Analyzer for Industrial Applications Using ARM and PIC," Int'l J. of Comp. Sci. and Info. Tech., 2012.

[9] N.N. Monot and B. Bavoux, "Fine-grained Simulation in the Design of Automotive Communication Systems," Embedded Real-Time Software and Systems (ERTSS 2012), Toulouse, France, 2012.

[10] "Real Time at Work: RTaW-Sim Brochure," Real Time at Work, Paris, 2014.