FlexRay Static Section Scheduling Using Full Model

Rim Bouhouch, Houda Jaouani, Wafa Najjar, Salem Hasnaoui

SysCom Laboratory
National Engineering School of Tunis
Tunis, Tunisia
{rim.bouhouch@yahoo.fr, jouani_houda@yahoo.fr, wafa_najjar@yahoo.fr, salem.hasnaoui@enit.rnu.tn}

***Abstract*—In this paper, we propose a new scheduling method for FlexRay static segment tasks, on the node level. This method handles the periodic communicating tasks transmitted on the static segment of the bus and takes into account the effect of the disruptive tasks, such as interruptions, on the response time. In this context, our scheduling method is based on the full model that we evaluate the performance by calculating the response time of the communicating tasks using as application model the SAE benchmark.**

Keywords-Scheduling; FlexRay Bus; Periodic Tasks; Full Model; Worst Case Response Time.

I. INTRODUCTION

FlexRay [1] is a new communication system that offers reliable and real-time capable high-speed data transmission between electrical and mechatronic components to map current and future innovative functions into distributed systems within automotive context. Thanks to its several features, this communication protocol is meeting safety critical applications performance requirements (flexibility, fault-tolerance, determinism, high-speed, etc.). Therefore, FlexRay is emerging as a predominant protocol for in-vehicle x-by-wire applications (i.e., drive-by-wire, steer-by-wire, brake-by-wire, etc.). As a result, there has been a lot of recent interest in timing analysis techniques in order to provide bounds for the message communication times on FlexRay. The real-time Data Distribution Service (DDS) based on the subscription-publication paradigm offers a clear distinction between the communicating tasks by classifying them into DataReaders and DataWriters and that helps insuring the delivery of the right data on the right time. One of the most interesting combinations would be the use of DDS on top FlexRay networks; but the challenge remains in the scheduling of the DataWriter provided by the applications and according to DDS specification, to meet the DataReaders Deadlines.

In this paper, we provide a scheduling method for periodic tasks on the static segment, based on the full model taking into account all the disruptive events and their effect on the response time of the Writers evaluated by the WCRT.

In the first section, we present the related work dealing with scheduling in the FlexRay bus; in the second section, an overview of the FlexRay network and its features is given; the third section is dedicated to scheduling parameters in the bus and in the static section; the fourth section presents the response time calculation using the full model; in the fifth section we present the application model on which we have performed our tests, and, in the last section, we present the results of our tests.

II. RELATED WORK

Tasks in real-time networks such as FlexRay [1] or CAN [14] are scheduled according to a static or a dynamic scheduling method. A static scheduler is a time triggered scheduling based on the Time Division Multiple Access (TDMA) [14], where each participant is granted a specific fixed interval in a repetitive time window. TDMA scheduling guarantees a deterministic transfer of messages, but has the disadvantage that the bandwidth is not used efficiently. A dynamic scheduling is an event triggered scheduling where participants can only send information if an event occurs, such as new data is ready for transmission.

Our previous researches [2] were interested in scheduling for the Data Distribution Service (DDS) architecture over CAN. We have developed in each node a local scheduling component, the Earliest Deadline First (EDF) scheduler. The latter, sends scheduling parameters of tasks to the global scheduling system. Then information is sent to a distributed information collection service called the System Information Repository (SIR). In [3], we have presented how DDS API is implemented on top of FlexRay Driver. In [4], we have presented a combined scheduling method that can be applied for both static and dynamic scheduling in FlexRay.

Related studies to this research include time triggered, event triggered and automobile protocols.

First studies [5] illustrate how a window-based analysis technique can be used to find Worst-Case Response time of a task. It considers bursty sporadic activities, where tasks arrive sporadically but then execute periodically for some bounded time.

Hagiescu *et al.* [6] proposes an analytical framework for compositional performance analysis of a network of Electronic Controller Unit (ECU) that communicates via a FlexRay bus. The main contribution was a formal model of the protocol governing the static segment of FlexRay.

In this paper, we focus our interest on the static segment of FlexRay and propose a new scheduling method that handles all the disruptive tasks and their effects on the response time, to evaluate the deadline of the communicating tasks.

III. FLEXRAY NETWORKS

FlexRay has been developed by the FlexRay consortium since 2000 for safety related applications in the automotive industry [1]. It is today applied in real-time application and as a replacement of CAN when higher data rates are required.

 ISBN: 978-1-61208-226-4

FlexRay has been developed to support x-by-wire applications such as steer-by-wire or brake-by-wire. These are replacements of the traditional mechanical and hydraulic control systems through electronic control systems.

FlexRay features two communication channels, each with a data rate of 10 Mbits/s, and payloads of frames up to 254 Bytes. Furthermore, the communication is time triggered in contrast to the event triggered CAN protocol. This is why FlexRay guarantees fixed communication latencies and a global synchronous time basis for all participating electronic control units.

A. Topologies

A FlexRay cluster consists of several nodes and two communication channels, channel A and channel B. In order to provide reliable communication, a node must be connected to both communication channels. To reduce cost using only one channel can be sufficient.

FlexRay supports both bus and star topologies. To increase the communication distance between two nodes they have to be connected via star couplers [7].

B. Hierarchical Network Timing

The communication scheme of a FlexRay cluster is built up of communication cycles that are repeated over again from startup of the network until it is shutdown. A communication cycle consists of the network communication time and the network idle time.

The communication time includes a mandatory static segment, an optional dynamic segment, and the symbol window.

In the static segment, deterministic communication ensures constant latency. FlexRay adheres to a time division multiple access method (TDMA), which means that there are equally sized slots and that the point of time is fixed when a frame is transmitted on the channel.

In the dynamic segment event driven communication takes place. This is usually used for low priority data, for example for the transmission of diagnosis information [8].

C. Electronic Control Unit (ECU)

The software application is executed on a host processor which is connected to a dedicated communication controller that executes the FlexRay protocol. The transmission from digital signals of the communication controller to analog signals on the bus is accomplished by the bus driver.

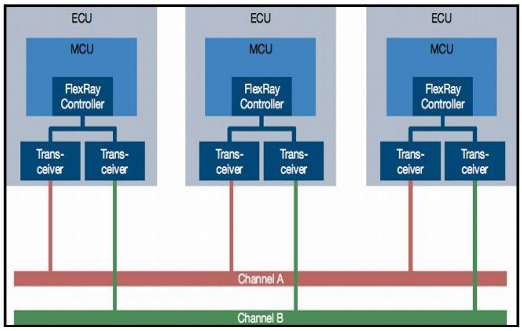

Figure 1. FlexRay Node (ECU)

IV. SCHEDULING PARAMETERS IN FLEXRAY NETWORKS

A. FlexRay Bus

FlexRay is a real time communication bus [1] designed to operate at speeds of up to 10 Mbits/s. He was developed by a consortium that includes automobile builders. It offers time-triggered and an event triggered architecture. Data is transmitted in payload segment containing between 0 and 254 bytes of data, 5 bytes for the Header segment and 3 bytes for the trailer segment. The topology may be linear bus, star or hybrid. This bus contains two channels; each node could be connected to either one or both channels.

FlexRay bus contains a static segment for time triggered messages and a dynamic segment for event triggered messages. In time triggered networks, nodes only obtain network access at specific time periods, also called time slots. In event triggered networks nodes may obtain network access at any time instant. The static (ST) segment and the dynamic (DYN) segment lengths can differ, but are fixed over the cycles. Both the ST and DYN segments are composed of several slots. The first two bytes of the payload segment are called message ID, this is used only in dynamic segment. The message ID can be used as a filterable data.

In this paper, we will study the transmission parameters of DDS nodes on a FlexRay bus. During any slot, only one node is allowed to send on the bus, and that is the node which holds the message with the frame identifier (Frame ID) equal to the current value of the slot counter. There are two slot counters, corresponding to the ST and DYN segments, respectively. The assignment of frame identifiers to nodes is static and decided offline, during the design phase. Each node that sends messages has one or more ST and /or DYN slots associated to it. The bus conflicts are solved by allocating offline one slot to at most one node, thus making possible for two nodes to send during the same ST and DYN slot. FlexRay allows the sharing of the bus among event driven (ET) and time driven (TT) messages.

For a distributed system based on FlexRay, task scheduling can be SCS (Static Cyclic Scheduling) or FPS (Fixed Priority Scheduling). For the SCS tasks and ST messages, the schedule table could be built. For FPS tasks and DYN messages, the worst-case response times had to be determined.

B. Communication Cycle

The FlexRay protocol organizes time into communication cycles, every cycle is organized into four parts, segments of configurable duration: The static segment is used to send critical, real-time data, and is divided into

 ISBN: 978-1-61208-226-4

static slots, in which the electronic control units (ECUs) can send a frame on the bus. These frames consist of a header, payload and trailer and are assigned to the slots according to a static, TDMA-based schedule. Channel idle time is enforced between frames to prevent overlapping consecutive frames. The dynamic segment enables event-triggered communication. The lengths of the mini slots in the dynamic segment depend on whether or not an ECU sends data. The symbol window is used to transmit special symbols, for example to start up the FlexRay cluster. The network idle time interval is used by the nodes to allow them to correct their local time bases in order to stay synchronized to each other.

The length of an ST slot is specified by the FlexRay global configuration parameter gdStaticSlot. The length of the DYN segment is specified in number of mini-slots gNumberOfMinislots.

C. Static segment parameters

In a general communication process, response time can be divided in four pieces, as shown in Fig. 1: generation delay, queuing delay, transmission delay and reception delay [9].

Generation delay is started when the transmitting node received the request of sending from a frame until the data is written into the buffer and ready for being sent. Queuing delay is started when generation delay ended until the frame acquires the occupation of the bus and begins to be sent. Transmission delay is the time during which the frame is being transmitted on the bus. Reception delay is started when the frame gets off the bus and goes into the receiving node until the frame accomplishes its task.

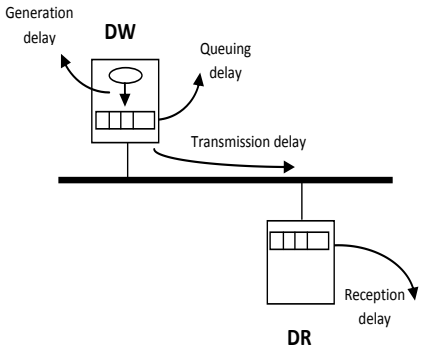

Figure 1. Communication Model between DataReader and DataWriter

Note that the generation delay and reception delay are not related to the FlexRay network characteristics, but related to the given MCU performance. Therefore, these two parts of delay should not be taken into account. In FlexRay protocol the average response time R_m of a given frame is the sum of queuing delay average (t_m) and transmission delay average (C_m):

$$R_m = t_m + C_m \quad (1)$$

Since the static segment is transmitting at fixed time points in each FlexRay communication cycle without any queuing delays, the response time can be approximated by Cm.

$$R_m = C_m \quad (2)$$

Transmission delay Cm refers to the time interval between being on the bus and completion of sending process. It depends on the frame itself as well as bus parameters.

$$C_{m,s} = [TSS + FSS + FES + t_d + (HS + TS + S_m) \times (8 + BSS)]\tau_{bit} \quad (3)$$

TSS is the Transmission Start Sequence (3~15 bits). FSS is the Frame Start Sequence (1 bit). FES is the Frame End Sequence (2 bits). t_d is the delay related to sending and receiving nodes, which is around 2~3 bits. S_m represents the data field length (number of bytes) of the data frames. In addition, two BSS (Bit Start Sequence) are added before each byte. The constant "8" added to the data field length Sm refers to the sum of the FlexRay Header Segment (HS: 5) and Trailer Segment (TS: 3) lengths (number of bytes). Finally, τ_{bit} refers to the one bit transmission delay.

V. RESPONSE TIME CALCULATION

The full model is inspired from the FPS (First Priority Scheduling) approach [10], which is the most widely used approach in the computing world. In this case, each task has a fixed static priority, which is ECU pre-run-time. The runnable tasks are executed in the order determined by their priority, knowing that in real-time systems, the "priority" of a task is derived from its temporal requirements, not its importance to the correct functioning of the system or its integrity.

The full model was conceived to be used in an industrial context [10], the temporal overheads of implementing the system must be taken into account such as:

- Context switches (one per job)
- Interrupts (one per sporadic task release)
- Real-time clock overheads

In this case, the Response time equation is rather than:

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \quad (4)$$

where $hp(i)$ is the set of tasks with priority higher than task i, C_i is the worst case computation time of the task i and T_j is the minimum time between task releases, jobs or task period.

The new equation is:

$$R_i = CS^1 + C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil (CS^1 + CS^2 + C_j) \quad (5)$$

where the new terms CS^1 and CS^2 are the cost of switching to the task, and the cost of switching away from it. And the term B_i is the cost of the task worst case blocking time.

The cost of handling interrupts is:

$$\sum_{k \in \Gamma_s} \left\lceil \frac{R_i}{T_k} \right\rceil IH \quad (6)$$

where Γ_s is the set of sporadic tasks and IH is the cost of a single interrupt (which occurs at maximum priority level).

There is also a cost per clock interrupt, a cost for moving one task from delay to run queue and a (reduced) cost of moving groups of tasks

 ISBN: 978-1-61208-226-4

Let CT_c be the cost of a single clock interrupt, $\Gamma_p$be the set of periodic tasks, and CT_s be the cost of moving one task. The following equation can be derived

$$R_i = CS^1 + C_i + B_i + \sum_{j \epsilon hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \left(CS^1 + CS^2 + C_j\right) + \sum_{k \epsilon \Gamma_s} \left\lceil \frac{R_i}{T_k} \right\rceil IH + \left\lceil \frac{R_i}{T_{clk}} \right\rceil CT_c + \sum_{g \epsilon \Gamma_p} \left\lceil \frac{R_i}{T_g} \right\rceil CT_s \quad (7)$$

Within the static segment a static time division multiple access scheme is applied to coordinate transmissions. In the static segment all communication slots are of identical, statically configured duration and all frames are of identical, statically configured length. In order to schedule transmissions each node maintains a slot counter state variable *vSlotCounter* for channel A and a slot counter state variable *vSlotCounter* for channel B. Both slot counters are initialized with 1 at the start of each communication cycle and incremented at the end boundary of each slot.

In the Implementations of the FlexRay bus, the periodic and safety-critical data is scheduled on the static time-triggered segment so the tasks in the static segment are periodic tasks that have the same priority per communication cycle.

Considering these facts the equation (7) applied on the static segment context becomes:

$$R_i = CS^1 + C_i + B_i + + \sum_{k \epsilon \Gamma_s} \left\lceil \frac{R_i}{T_k} \right\rceil IH + \left\lceil \frac{R_i}{T_{clk}} \right\rceil CT_c + \sum_{g \epsilon \Gamma_p} \left\lceil \frac{R_i}{T_g} \right\rceil CT_s \quad (8)$$

VI. APPLICATION MODEL

To illustrate the utility of our Comprehensive Scheduling Strategy, we have chosen to work within a platform of a vehicular network based on the SAE standard. In this system, a set of network processors subsystems produces routing data. This data must be distributed along the vehicular network.

In fact, we will apply the studied approaches on a new vehicle benchmark developed in [11] and based on the SAE Benchmark [15]. We added to the original benchmark a number of nodes and messages to better represent the complexity of today's vehicles and to model some added options responsible for improving vehicle safety, reliability, cost, and luxury.

However, this Benchmark was designed to best fit the CAN network and it needs major modifications to be adapted to the FlexRay protocol. Hence, later in this paper, we will explain how to introduce adjustments to that model and we will apply our scheduling algorithm and present our results for the new model. The resulting architecture is composed of 15 nodes connected by the FlexRay bus. According to the FlexRay specification, each node consists of a host (CPU) that processes incoming messages and generates outgoing messages, a communication controller (CC) that independently implements the FlexRay protocol services, and a two-way controller-host interface (CHI) that serves as a buffer between the host and the CC.

The main goal of the proposed architecture is to insure better performance of the vehicular network and to guarantee the arrival of the right data on the right time by meeting the tasks deadline. The framework architecture is a set of nodes connected via FlexRay Real-Time Transport protocol. In each node is embedded a Real-Time Operating System µCOSII and a publish/subscribe middleware.

VII. RESULTS AND COMMENTS

In this section, we propose an algorithm to calculate the response time of the DataWariters tasks.

The equation (8) gives us the needed parameters to determine the response time for both static and dynamic segments tasks:

- C_i the computing time is equivalent to the transmission delay $C_{m,s}$ and $C_{m,d}$, because the execution of a message relative to a writing task is the fact to transmit data on the bus.
- The worst blocking time Bi is defined as follows:

$$B_i = \frac{Frame\ size - 1\ bit}{Lowest\ flow\ rate\ used} \quad (9)$$

This equation is true for the CAN case; but. in the FlexRay case:

$$B_i = 0$$

- CS^1 is the cost of switching to the task. This parameter is given by the used real-time operating system µCOSIII [12].

$$CS^1 = 0.005\ ms$$

- CS^2 is the cost of switching away from the task, this parameter is also given by the used real-time operating system µCOSIII [12].

$$CS^2 = 0.009\ ms$$

- IH is the cost of executing an interrupt service routine. This interrupt is supposed to be at the maximum priority level. The number of STATUS registers present in the system determines the time taken by the handler to execute the interrupt routine. The FlexRay driver interrupt routine takes more time in response to the status of receiving communications data. For our study we approximate this parameter as follow:

$$IH = 10 \times CT_c$$

- CT_c is the cost of a single clock interrupt for the microcontroller MB91F465X we have approximated its value:

$$CT_c = \frac{1}{10} \times T_{clk}$$

- CT_s is the cost of moving one task, which is equivalent to switching a task.

$$CT_s = CS^2$$

- T_{clk} is the clock period calculated for a given core frequency.

The response time calculation process is described by the following algorithm:

Algorithm Worst Case Response Time Computing

```
for i in 1..N loop
  n := 0
  loop
    Calculate Ci for periodic tasks
    Calculate Ci for sporadic tasks
    n := n + 1
  end loop
```

```
end loop
for i in 1..N loop
  n := 0
  W_i^n = C_i
  loop
    calculate new w_i^(n+1)
    if w_i^(n+1) = w_i^n  then  R_i = w_i^n
      exit value found
    end if
    if w_i^(n+1) > T_i  then
      exit value not found
    end if
    n := n + 1
  end loop
end loop
```

For the simulation, we consider a set of FlexRay nodes the sending 36 messages on the FlexRay bus. Since each node in the system that generates static messages needs at least one static slot, the minimum number of static slots is the number of nodes (*nodesST*) sending static messages [1].

In the extended benchmark [11], there are 15 nodes sending 36 messages; among them, 30 are periodic messages that need to be scheduled on the FlexRay static segment. We will regroup these nodes into 6 for the simulations.

The period of the bus cycle (*gdCycle*) must be lower than the maximum cycle length *cdCycleMax* equal to 16 ms and has to be, also, an integer divisor of the period of the global static segment. In addition, each node has a counter *vCycleCounter* in the interval 0…63. Thus, during a period of the global static schedule there can be at most 64 bus cycles. Observing our message set, we have noticed that almost all of the message periods are multipliers of 5 ms. So we can fix the period of the bus cycle to 5 ms and adjust some message periods, especially the messages introduced by Ben Gaid, M-M in [13] and others introduced by M. Utayba in [11].

All messages with period equal to 8 ms will have a new period of 5 ms, and the messages with period equal to 12 ms will have a period of 10 ms. This will not affect our system efficiency since it will make it faster and more reactive.

There is another problem with messages having a 1000 ms period; they cannot be scheduled with a bus cycle of 5ms and 64 cycles. In fact, even if we consider the longest period of the global static schedule (64 bus cycles), we wouldn't manage to reach the 1000 ms. Thus, we have to decrease this period to 64*5=320 ms.

We have also replaced the original bus priorities designed for an event triggered bus (CAN) by a local priority able to order transmission of messages having the same Frame Identifier on different slots assigned to their source node.

Applying the previous algorithm with a bus speeds of 10 Mbit/s for one channel transmission scheme, and a core frequency of 12 Mhz. The results obtained are summarized in Table I.

TABLE I. BODY CONTROL MODULE RESULTS

Vehicle Module	Message ID	Size (Bytes)	Deadline [ms]	T [ms]	Task Priority	Worst Case Response Time R (ms)
BODY Control Module	3	1	5	5	1	0.1397
	13	1	**5**	**5**	1	0.1397
	31	4	100	100	1	0.1487
	34	3	**320**	**320**	1	0.1457
Engine Controller Module	4	2	5	5	1	0.1424
	6	2	5	5	1	0.1424
	20	2	10	10	1	0.1424
	35	1	**320**	**320**	1	0.1394
Active Suspension Unit	27	2	**10**	**10**	1	0.2229
Active Frame Steering	22	2	10	10	1	0.2229
Electronic Brake Control Module	14	4	**5**	**5**	1	0.2289
Traction Control Unit	8	1	5	5	1	0.2199
	15	4	**5**	**5**	1	0.2289
ESP/ROM	16	4	**5**	**5**	1	0.2289
	28	5	**10**	**10**	1	0.2319

 ISBN: 978-1-61208-226-4

Hydraulic Brake Control Unit	2	2	5	5	1	0.2499
	32	1	100	100	1	0.2469
Transmission Control Unit	5	1	5	5	1	0.2469
	33	1	100	100	1	0.2469
	36	1	**320**	**320**	1	0.2469
Throttle Control Unit	7	1	5	5	1	0.2469
Adaptive Cruise Control	29	3	10	10	1	0.2529
Front-Right Wheel Module	10	1	**5**	**5**	1	0.1389
	24	2	**10**	**10**	1	0.1419
Rear-Right Wheel Module	12	1	**5**	**5**	1	0.1389
	26	2	**10**	**10**	1	0.1419
Front-Left Wheel Module	9	1	**5**	**5**	1	0.1389
	23	2	**10**	**10**	1	0.1419
Rear-Left Wheel Module	11	1	**5**	**5**	1	0.1389
	25	2	**10**	**10**	1	0.1419

We notice, on this table of results, that the entire tasks deadline is matched. For the worst case response time using the worst case Core frequency which is 12 Mhz, we have noticed that the deadline has been met and the equation below is verified.

$$D \geq R$$

Thanks to FlexRay bus speed, we can assume that the DDS Deadline QoS Policy can always be reached.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we have proposed to use DDS on top of the real-time network FlexRay to take advantage of its high speed and to profit of the DDS QoS management in an automotive context. We have proposed a scheduling model based full FPS scheduling to first calculate the worst case response time for our vehicular system and evaluate its performance on a benchmark application, an extended SAE benchmark. After the simulations, results have shown that the applications deadline requirements have been met. One promising research direction would be the evaluation of the real-time QoS parameters offered by DDS on the same system configuration.

ACKNOWLEDGMENT

The researches presented in this paper would not have been possible without the support of many people. We wish to express our gratitude to the SYSCOM ENIT members for their help and assistance.

REFERENCES

[1] FlexRay Consortium," FlexRay Communications System-Protocol Specification", Version 2.1, Revision A, 2005.

[2] T. Guesmi, R. Rekik, S. Hasnaoui, and H. Rezig, "Design and Performance of DDS-based Middleware for Real-Time Control Systems", IJCSNC, vol. 7, No. 12, 2007, pp. 188-200.

[3] R. Bouhouch, W. Najjar, H. Jaouani, and S. Hasnaoui, "Implementation of Data Distribution Service Listeners on Top of FlexRay Driver", INFOCOMP 2011, IARIA, October 2011,Barcelona Spain, pp. 64-69.

[4] W. Najjar, R. Bouhouch, H. Jaouani, and S. Hasnaoui, "Static and Dynamic Scheduling for FlexRay Network Using the Combined Method", International Journal of Information Technology and Systems, Vol. 1, No. 1, January 2012, pp. 18-26.

[5] K. W. Tindell, A. Burns, and A. J. WELLINGS," An extendible approach for analyzing fixed priority hard real-time tasks", Real_Time Systems, Vol. 6, No. 2, 1994, pp. 133-151

[6] A. Hagiescu, U. Bordoloi, and S. Chakraborty, "Performance Analysis of FlexRay-based ECU Networks", DAC proceedings, 2007, pp. 284-289

[7] M. Gerke, "FlexRay : A state of the art vehicle bus, Embedded Systems Lecture", Chair Professor Finkbeiner Saarbrucken, Dec 2008, pp. 3-4.

[8] A. Zhao, "Reliable In-Vehicle FlexRay Network Scheduler Design", master of science thesis in Electrical Engineering, Mai 2011, Delft University of Technology,The Netherlands, pp. 17-23.

[9] T. Guangyn, B. Peng, and C. Quanshi, "Response Time Analysis of FlexRay Communication in Fuel Cell Hybrid Vehicle", Vehicle Power and Propulsion Conference VPPC'08. IEEE, 2008, pp. 1-4.

[10] A. Burns and A. Wellings, "Scheduling Real-Time Systems", Chapter 11, Real-Time Systems and Programming Languages, The university of York, Department of Computer Science, pp. 121-125.

[11] M. Utayba and N. Al-Holou, "Development of An Automotive Communication Benchmark", Canadian Journal on Electrical and Electronics Engineering, Vol. 1, No. 5, August 2010.

[12] A J. J. Labrosse. "MicroC/OS-II The Real Time Kernel". Miller Freeman, Inc, United States of America, 1999.

[13] M-M. Ben Gaid, A. Cela, S. Diallo, R. Kocik, R. Hamouche, and A. Reama, "Performance Evaluation of the Distributed Implementation of a Car Suspension System", the IFAC Workshop on Programmable Devices and Embedded Systems, February 2006, pp.776-787 .

[14] K. Tindell and A. Burns, "Guaranteeing Message Latencies on Control Area Network (CAN)", Real-Time Systems

 ISBN: 978-1-61208-226-4

Research Group, Department of Computer Science, University of York, England, pp. 5-6.

[15] H. Kopetz, "A solution to an automotive Control System Benchmark", Real-Time Systems Symposium, 7-9 Dec. 1994, pp. 154-158. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.110.3545&rep =rep1&type=pdf.

 ISBN: 978-1-61208-226-4