Enhancement of Controller Area Network (CAN) Bus Arbitration Mechanism

Chin-Long Wey, Chung-Hsien Hsu, Kun-Chun Chang*, and Ping-Chang Jui**

Department of Electrical Engineering, National Chiao Tung University, Hsinchu, Taiwan
**Deparment of Electrical Engineering, National Central University, Jhongli, Taiwan*

***Abstract*—FlexRay protocol includes a static time-triggered and a dynamic event-triggered segment, where the *static segment* enabling a guaranteed real-time transmission of critical data, while the *dynamic segment* (optional) for low-priority and event-triggered data. FlexRay is operated in a synchronous fashion. Based on the CAN (Controller Area Network) bus arbitration mechanism, this study takes the most significant bit (MSB), D10, of the identifier to assign the messages to be either static-like or dynamic-like, where the static-like type has higher priority than the dynamic-like type. Both types are operated in an asynchronous fashion as the conventional CAN protocol. With the designers' experience with good schedulability, the above assignment can ensure that all critical messages be sent out before their deadline and no one message type hugs the bus. However, the salient feature is achieved at the cost of losing one bit for the identifier.**

Keyword: CAN Bus, Arbitration Mechanism, FlexRay, Static Segment, Dynamic Segment.

I. Introduction

CAN (Controller Area Network) bus is a vehicle bus standard designed to allow microcontrollers and devices to communicate with each other within a vehicle without a host computer [1]. It is a message-based protocol for the applications of automotive, industrial automation, medical equipment, and etc. A modern automobile may have as many as 70 electronic control units (ECUs) for its subsystems, such as transmission, powertrain, antilock braking system, airbag, electric power steering, battery and recharging system. A subsystem may need to control actuations or receive feedback from sensors. The communications among these subsystems are essential.

CAN bus is a multi-master broadcast serial bus standard for connecting ECUs. Each node is able to send and receive messages, but not simultaneously. A message consists primarily of an ID (identifier) which represents a CAN node [2]. It is transmitted serially onto the bus. The signal pattern is encoded in NRZ (Non-return-to-zero) and sensed by all nodes. If the bus is free, any node may transmit data. If two or more nodes simultaneously begin sending messages, however, the sender with higher priority will send and the reminding one are held. More specifically, the message with the more dominant ID will overwrite other nodes with less dominant IDs, so that eventually only the dominant message remains and is received by all nodes. This mechanism is referred to as *priority based bus arbitration* [1-5]. Messages with numerically smaller values of IDs have higher priority and transmitted first.

When used with a differential bus, a carrier sense multiple access/bitwise arbitration (CSMA/BA) scheme is often implemented: if two or more devices start transmitting at the same time, the priority based arbitration scheme decides which one will be granted permission to continue transmitting. The prioritized arbitration solution makes CAN very suitable for real time prioritized communications systems. CAN provides nondestructive bus arbitration because the highest priority message doesn't get destroyed.

The goal is to ensure that the highest-priority work gets completed as soon as possible. At the same time, a high-priority job should never miss its deadline because it was waiting for a lower-priority task to complete. If a number of nodes compete, at each process cycle, the highest-priority node among them will win out, and the remaining nodes will try again in the next cycle. As a result, the process will lock out lower-priority messages. In order to ensure that no one message type hugs the bus, this study presents an alternative bus arbitration mechanism to assist the designers for developing the control strategy.

The FlexRay bus is the prospective automotive standard communication system [6-12]. For the sake of a high flexibility, the protocol includes a static time-triggered and a dynamic event-triggered segment. The *static segment* enabling a guaranteed real-time transmission of critical data, while the *dynamic segment* (optional) for low-priority and event-triggered data. Both types are operated in synchronous fashion. The CAN bus architecture does not impose any restrictions on when nodes are allowed to place messages on the bus, while FlexRay can pre-allocate time slots for the messages. In fact, it is possible to have CAN networks operating at near 100% bus bandwidth [2], and the FlexRay adapting the TDMA (Time Division Multiple Access) scheme may send the null data which wastes the slot time and space. In addition, FlexRay requires a precise oscillator clock to synchronize the operations [5,6].

Taking the advantages of simpler and less expensive CAN bus structure; this study is to implement the design concept of static and dynamic segment message allocation scheme in an asynchronous fashion to the event-driven CAN bus ensuring that no one message type hugs the bus. More specifically, the messages are classified as critical and non-critical based on its importance. When non-critical messages will be sent, they are assigned to the dynamic-like segment, and the critical messages are assigned to the static-like segment. At the application layer, based on the designers' experiences with good schedulability [13,14], the MCU (Micro-Controller Unit) of each node determines the messages to be critical or non-critical and assigned them to the event-driven static-like or dynamic-like segment. For simplicity of discussion, both static-like or dynamic-like segments for CAN bus are called *static_C* and *dynamic_C* segments, *SCS* and *DCS* segments, for short, respectively.

For a standard data frame of CAN, the arbitration field consists of 12 bits—11 identifier bits, ID10~ID0, and 1 RTR (Remote Transmit Request) bit [1]. In this implementation, the bit ID10 (the bit next to SOF, Start of Frame) is used to determine the *SCS* or *DCS* segment. Results of this study will show that with the designers' experiences with good schedulability, the proposed bus arbitration scheme can ensure that no one message type hugs the bus.

The next section will briefly review the basic CAN bus arbitration mechanism and the FlexRay's segmentation scheduling scheme. Section III presents the proposed CAN bus arbitration mechanism with some examples. Finally, a concluding remark is briefly given in Section IV.

II. Preliminary

This section briefly reviews the CAN arbitration mechanism and the dynamic and static segments in FlexRay.

A CAN node, as shown in Fig. 1, is comprised of a micro controller unit (MCU), CAN controller, and CAN transceiver. In this experiment, 8051, SJA1000, and PCA82C250 are used as the micro-controller, CAN controller, and CAN transceiver, respectively, for each CAN node.

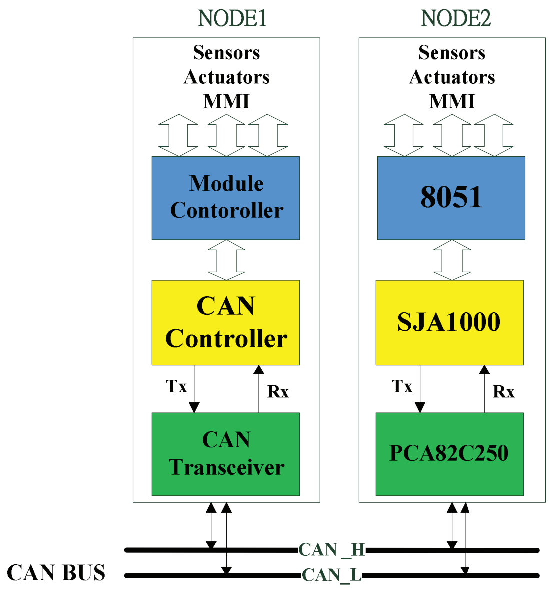


Fig. 1 Typical CAN Node Implementation.

2.1 CAN Arbitration Mechanism

A CAN message transmitted with highest priority will succeed and the node transmitting the lower priority message will sense this and back off and wait [2]. If one node transmits a dominant bit and another node transmits a recessive bit, then the dominant bit wins. So, if a recessive bit is being transmitted while a dominant bit is sent, the dominant bit is displayed, evidence of a collision. A dominant bit is asserted by creating a voltage across the wires while a recessive bit is simply not asserted on the bus. If any node sets a voltage difference, all nodes will see it. Thus, there is no delay to the higher priority messages, and the node transmitting the lower priority message automatically attempts to re-transmit six bit clocks after the end of the dominant message.

Arbitration is performed during the transmission of the identifier field. Each node starting to transmit at the same time sends an ID with dominant as binary 0, starting from the high bit. As soon as their ID is a larger number (lower priority), they will be sending 1 (recessive) and see 0 (dominant), so they back off. At the end of ID transmission, all nodes but one have backed off, and the highest priority message gets through unimpeded.

For example, consider an 11-bit ID CAN network, with two nodes with IDs of 15(00000001111), namely, D15, and 16(00000010000), D16. If these two nodes transmit at the same time, each will transmit the first six zeros of their ID with no arbitration decision being made. When the 7th bit is transmitted, the node D16 transmits a 1 (recessive) for its ID, and the node D15 transmits a 0 (dominant). When this happens, the node D16 will realize that it lost its arbitration, and allow the node D15 to continue its transmission. This ensures that the node with the lower bit value will always win the arbitration. The ID with the smaller number will win the right to use.

2.2 FlexRay and TTCAN

FlexRay [6], as a communication protocol for automotive control system, is developed to fulfill the increasing demand on the electronic control units for implementing systems with higher safety and more comfort. Fault-tolerant feature is especially highlighted in the FlexRay protocol such that it is robust enough for applying to the safety-critical applications.

For the sake of a high flexibility, the protocol includes a static time-triggered and a dynamic event-triggered segment. The *static segment* enabling a guaranteed real-time transmission of critical data, while the *dynamic segment* (optional) for low-priority and event- triggered data.

The fixed time slots are situated in the static segment at the beginning of a bus cycle. In the dynamic segment the time slots are assigned dynamically.

The TTCAN (time-triggered CAN) protocol is a higher layer protocol on top of the CAN data link layer. Time-triggered communication means that activities are triggered by the elapsing of time segments. In a time-triggered communication system all points of time of message transmission are defined during the development of a system [15]. A time-triggered communication system is ideal for applications in which the data traffic is of a periodic nature.

The time-triggered control and thus the synchronization of the involved control units within the network are done via a reference message. As soon as SOF is recognized, the local time unit is synchronized. In this protocol, it is required an oscillator to precisely generate the clock for synchronization.

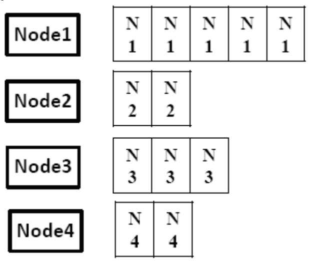
For some applications it is necessary that the reference frame is sent corresponding to an event. For these cases a bit is inserted in the basic cycle, which discontinues the periodic transmission of messages until the event has taken place.

III. Proposed Arbitration Mechanism

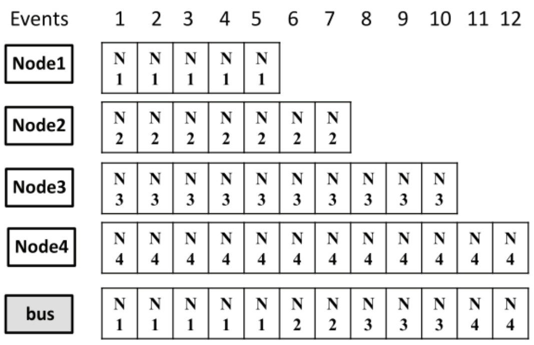
This section presents some observations of the CAN arbitration mechanism and describes the modification to enhance the arbitration capability of typical CAN bus.

3.1 Arbitration Mechanism

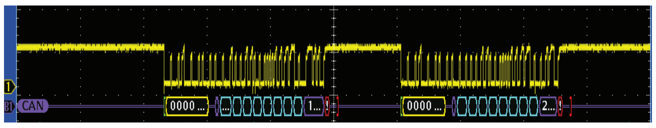
Based on the CAN arbitration mechanism described previously, the node with the highest priority may hug the bus until it completely sends all the messages output. For example, consider there are 4 nodes to simultaneously send their messages to the bus. Without loss of generality, we assume that Node1 has the highest priority, Node2 is the second, Node3 is the third, and Node4 has the lowest priority. Fig. 2(a) shows the messages held in each node. Fig. 2(b) illustrates the arbitration process. At the first cycle, Node1 with the highest priority wins out and send the first message to the bus. There's nothing to stop the highest-priority node from being transmitted again. Thus, Node1 continually sends its messages until all of its messages are sent. Each frame contains 8 bytes data. Node1 may send data which are included in 5 frames. Then, this is followed by sending the messages in Node2 during the cycle 6 and 7. It takes turns to complete the messages in Node3 and Node4, respectively.



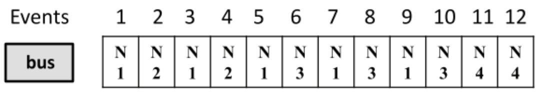
(a)


Events 1 2 3 4 5 6 7 8 9 10 11 12
Node1
N 1 N 1 N 1 N 1 N 1
Node2
N 2 N 2 N 2 N 2 N 2 N 2 N 2
Node3
N 3 N 3 N 3 N 3 N 3 N 3 N 3 N 3 N 3 N 3
Node4
N 4 N 4 N 4 N 4 N 4 N 4 N 4 N 4 N 4 N 4 N 4 N 4
bus
N 1 N 1 N 1 N 1 N 1 N 2 N 2 N 3 N 3 N 3 N 4 N 4

(b)



(c)


Events 1 2 3 4 5 6 7 8 9 10 11 12
bus
N 1 N 2 N 1 N 2 N 1 N 3 N 1 N 3 N 1 N 3 N 4 N 4

(d)

Fig. 2 Arbitration Mechanism: (a) Node Contents; (b) Arbitration Process, (c) Frame; and (d) Due to Message Setup Time.

When a message transmits, the node first loads the message identifier, data bytes, and control bits into the transmit message assembly registers. The node then transfers the data to the CAN protocol engine. The protocol engine creates the actual frame by inserting the frame elements, such start and stop bits and interframe space bits. The protocol engine also handles bus arbitration, cyclical redundancy check sum calculations, and looks for transmission errors [1]. Thus, when a message is being sent from a node, it is required time for preparing the next message to send. There exist some idle times between sending two consecutive messages.

In our experiments, a typical CAN node is described in Fig. 1, where 8051, SJA1000, and PCA82C250 are used as the micro-controller, CAN controller, and CAN transceiver, respectively. Fig. 2(c) illustrates two consecutive messages of same node are being sent. Once the first message is completely sent, it takes time to set up the second message. The time required for setting up the message is referred to as *message set up time*. In this experiment, the message setup time was measured as 90 μs. Because of the message setup time, it is possible that the second highest-priority node can take place during the period of the message setup time for the higher-priority node which has not yet been ready. The example arbitration process in Fig. 2(a) may result in a sequence shown in Fig. 2(d), due to the message setup time. The resultant sequence shows that the highest-priority Node1 needs to wait until the 9th cycle to complete sending all its messages. In this case, Node1 may miss its deadline for sending the messages.

It's the designer's responsibility to ensure that no one message type hugs the bus [2], and also the higher-priority nodes do not miss their deadline. The Identifier field of the original CAN format is modified in this study.

3.2 Identifier Field

When data is transmitted over a CAN bus no individual nodes are addressed. Instead, the message is assigned an identifier (ID) that works as a unique tag on its data content [1,2]. The ID not only defines the message content but also the message priority.

A message in the standard format, as shown in Fig. 3, begins with the start bit SOF. This is followed by the arbitration field which contains the ID of the CAN format and is used to arbitrate access to the bus. Also, part of the arbitration field is the RTR (remote transmission request) bit which indicates whether the frame is request format (without any data, this type of message is used to trigger a transmission by another node) or a data frame.

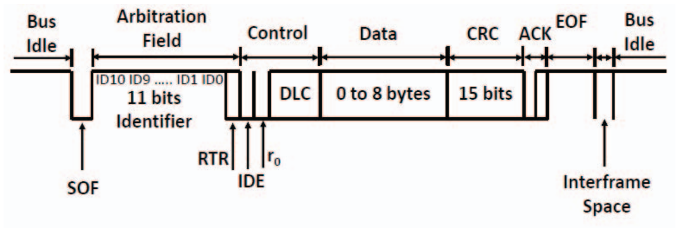


Fig. 3 Standard Format of CAN.

For a standard data frame of CAN, the arbitration field consists of 12 bits—11 identifier bits, ID10~ID0, and 1 RTR (Remote Transmit Request) bit. In this implementation, the bit ID10 (the most significant bit of the identifier) is used to set a logical "0" for the SCS segment and a logical "1" for the DCS segment. This implies that SCS segment has higher propriety than the DCS segment during the same cycle.

Based on the designers' experiences with good schedulability, and also on the sensor of the node transceiver and the corresponding actuators, the MCU at each CAN node determines the status of messages to be critical or not. The critical one will be assigned to the SCS segment; otherwise, the messages will go the DCS segment. In addition, the number of messages in the SCS segment should also be limited in order to prevent the highest-priority node from hugging the bus.

Based on the applications and the schedulability analysis, the designers can set the limits for the numbers of messages in both SCS and DCS segments can be continually sent. As illustrated in Fig. 4, once the number of messages in SCS segment, i.e., the number of S-cycles, exceeds the limit, the remaining messages in SCS segment are then changed to the DCS segment.

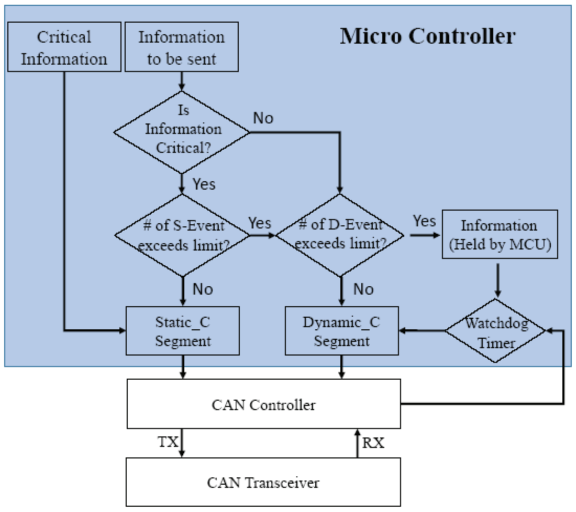


Fig. 4 Flow Chart of Segment Assignment.

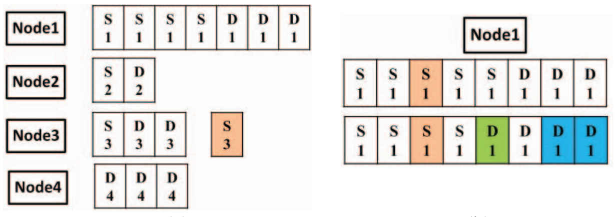
Further, once the number of messages in the DCS segment, referred to as the number of D-cycles, exceeds the limit, the MCU will hold these messages. A watchdog timer is used in MCU to count down the timer to zero and if the bus is still free, the messages held by the MCU are then, in turns, sent. Note that the emergent messages are defined as the critical messages which will be directly assigned to the SCS segment by the MCU and not in the loop of counting the limited number of messages in the SCS segment for that node. In general, the reason why the numbers of messaged in SCS and DCS segment exceed the limits is simply because the designers do not plan well. With the schedulability analysis, one can easily define both limits.

Consider the messages held in each node, as shown in Fig. 5(a). Again, Node1 has the highest priority, while Node4 is the lowest one. Suppose that two emergent messages were injected, one to Node1 during the 2nd event and the other one to Node3 during the 8th events. Both messages are assigned to SCS segments, as shown in Fig. 5(b) and colored in pink.

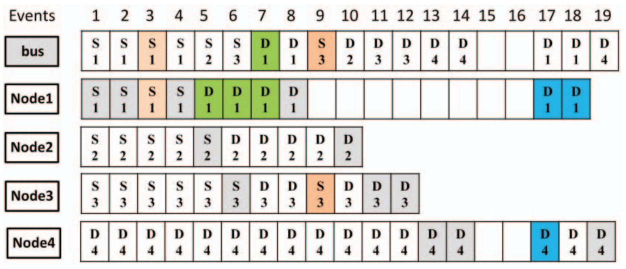
In this application, assume that the limits for the number of messages in SCS and DCS segments are 3 and 2 events, respectively, i.e., the limits of both S-event and D-event are 3 and 2, respectively. For Node1, the 4th SCS message (S1) will then be assigned as the DCS one, colored in green. Since the limit of D-event is 2, the 2nd and 3rd "D1" in Fig. 5(b) are then held by the MCU, colored in blue. Once the 2nd D1 is held, the following S1 has the highest priority to win the arbitration and transmit its message to the bus.

For simplicity of discussion, we assume that there exist no message setup times between two consecutive message transmissions.

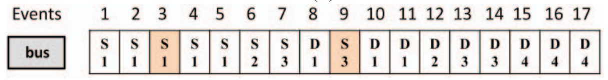
Fig. 5(c) illustrates the arbitration process. The first two S1 have the highest priority to send their messages to the bus. Then, the emergent message S1, colored in pink, also has the highest priority to send its message. Since the limit of S-event is 3, after sending the 3rd S1, the 4th S1 is then modified as D1, colored in green. At this event, the first S2 has the highest priority to send it message. It is followed by sending S3 at the 6th event. At the 7th event, the D1 (modified from S1) wins the arbitration and send its messages. It is followed by sending the next D1. At the 9th event, the emergent message S3 has the higher priority to send its message.



(a) (b)


Events
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
bus
S1 S1 S1 S1 S2 S3 D1 D1 S3 D2 D3 D3 D4 D4 D1 D1 D4
Node1
S1 S1 S1 S1 D1 D1 D1 D1 D1 D1
Node2
S2 S2 S2 S2 S2 D2 D2 D2 D2 D2
Node3
S3 S3 S3 S3 S3 S3 D3 D3 S3 D3 D3 D3
Node4
D4 D4 D4 D4 D4 D4 D4 D4 D4 D4 D4 D4 D4 D4 D4 D4 D4

(c)



(d)

Fig. 5 Proposed Arbitration Mechanism Enhancement: (a)&(b) Node Contents; (c) Arbitration Process, and (d) Better Planning.

Since the limit of D-event is 2, the next two D1 are then held by the MCU, colored in blue. At the 10th event, D2 sends its message, followed by sending two D3 and two D4. The last D4 is also held by the MCU, colored in blue. By the flow chart in Fig. 4, the watchdog timer is set to 500us. After the timer is counted down to zero and the bus is free, the messaged held by the MCU are then sent in the order of D1, D1, and D4 in the following 3 events. It takes a total of 19 events to complete all messages from these 4 nodes.

The example has shown that the SCS message can be sent anytime and the 6 critical messages were sent within 7 events (excluding the emergent messages) which do not miss the deadline.

However, if the designers have better planning by setting the limits of S-event and D-event to be 4 and 3 cycles, respectively, as illustrated in Fig. 5(c), no messages will be held by MCU and the watchdog timer is not needed for counting down.

IV. Conclusion

This study presents an alternative arbitration mechanism for CAN bus. Traditionally, when two or more nodes start transmitting at the same time, there is a priority based arbitration scheme to decide which one will be granted permission to continue transmitting. During arbitration, each transmitting node monitors the bus state and compares the received bit with the transmitted bit. If a dominant bit is received when a recessive bit is transmitted then the node stops transmitting. As a result, the messages at the highest-priority node will always hug the bus until all its messages are sent out and there is nothing to stop its transmissions.

In this study, the most significant bit, D10, of the identifier is used to assign to the SCS or DCS segment. At the application layer, based on the designers' knowledge and experiences, the MCU of each node determines the messages to be critical or non-critical and assigned them to the event-driven SCS or DCS segment in an asynchronous fashion. Results show that no one message type hugs the bus. With better planning and schedule strategy, all critical messages can be sent before their deadline. However, salient feature is achieved at the cost of losing one bit at the identifier.

Acknowlegement

This work was supported in part by the Taiwan National Science Council under the projects No. 102-2220-E-009-052 and 101-3113-P-110-004, and in part by Metal Industries Research Development Centre (MIRDC) and Ministry of Economic Affairs, Taiwan, under the project No. 102-EC-17-A-01-1010.

References

[1] *CAN specification Version 2.0*, Robert Bosch GmbH, 1991.
[2] M. Farsi, K. Ratcliff, and M. Barbosa, "An Overview of Controller Area Network." Journal of Computing & Control Engineering, pp.113-120, 1999.
[3] G.-S. Feng, W. Zhang, S.-M. Jia, and H.-S. Wu, "CAN Bus Application in Automotive Network Control," Proc. of International Conference on Measurement Technology and Mechatronics Automation, pp. 779-782, March 2010.
[4] R. Li, J. Wu, H. Wang, and G. Li, "Design Method of CAN BUS Network Communication Structure for Electric Vehicle," Proc. of International Forum on Strategic Technology, pp.326-329, October 2010.
[5] S.C. Talbot and S. Ren , "Comparison of FieldBus Systems, CAN, TTCAN, FlexRay and LIN in Passenger Vehicles" IEEE International Conference on Distributed Computing Systems, pp. 26-31, 2009.
[6] *FlexRay Communication System Electrical Physical Layer Specification Version 2.1 Revision B*. FlexRay Consortium, 2006.
[7] C.-C. Wang, G.-N. Sung, P.-C., Chen, and C.L. Wey, "A Transceiver Frontend for Electronic Control Units in FlexRay-based Automotive Communication Systems," IEEE Trans. on Circuits and Systems, I: Regular Papers, Vol. 57, No. 2, pp.460-470, February 2010.
[8] K.-L. Leu, Y.-Y. Chen, C.L. Wey, and J.-E. Chen, "Robustness Investigation of the FlexRay System," Proc. of IEEE Symposium on Industrial Embedded Systems, Lausanne, Switzerland, July 2009.
[9] K.-L. Leu, Y.-Y. Chen, C.L. Wey, and J.-E. Chen, "Robustness Analysis of the FlexRay System through Fault Tree Analysis," Proc. of IEEE International Conference on Vehicular Electronics and Safety (ICVES 2010), Shandong, China, July 2010.
[10] K.-L. Leu, Y.-Y. Chen, C.L. Wey, J.-E. Chen, and C.-H. Huang, "A Bayesian Network Reliability Modeling for FlexRay Systems," Proc. of International Conference on Information and Communication Technologies (ICICT 2010), Tokyo, Japan, May 2010.
[11] K.-L. Leu, Y.-Y. Chen, C.L. Wey, and J.-E. Chen, "A Verification Flow for FlexRay Communication Robustness Compliant with IEC 61508," Proc. of IEEE 2nd International Conf. on Industrial Mechatronics and Automation (ICIMA), Wuhan, China, May 2010.
[12] K.L. Leu, Y.Y. Chen, C.L. Wey, J.E. Chen and C.H. Hsu "A Bayesian Network Reliability Modeling for FlexRay Systems" World Academy of Science, Engineering and Technology, pp.42-47, 2000.
[13] R.I. Davis, A. Burns, R.J. Bril, and J.J. Lukkien, "Controller Area Network (CAN) Schedulability Analysis: Refuted, Revisited and Revised". Real-Time Systems, Volume 35, Number 3, pp. 239-272, April 2007.
[14] S. Mubeen, J. Maki-Turja, and M. Sjodin, "Extending Schedulability Analysis of Controller Area Network (CAN) for Mixed (Periodic/ Sporadic) Messages," Proc. of the 16th IEEE Conference on Emerging Technologies and Factory Automation (ETFA), Toulouse, France, 2011.
[15] Time-Trigger CAN: http://www.can-cia.org/index.php?id=166.