24

Advanced Bit Stuffing Mechanism for Reducing CAN Message Response Time

Kiejin Park and Minkoo Kang
Ajou University
Republic of Korea

1. Introduction

As customer requirements for safety and convenience in automobiles increases, so does the quantity of electronics and software installed in them. The amount of signal data from the electronics systems needs to be managed, making the design of communication protocols for in-vehicle networks (IVN) more important (Navet et al., 2005). The IVN protocols can be classified into two paradigms: event-triggered and time-triggered (Obermaisser, 2004). The event-triggered protocols are efficient in terms of network utilization, because the messages transmitted within event-triggered protocols are only transmitted when specific events occur. This differs from time-triggered communication in that the response time of message transmission is not predictable (Fabian & Wolfgang, 2006).

The controller area network (CAN) is a well-known event-triggered protocol originally developed in the mid-1980s for multiplexing communication between electronic control units (ECUs) in automobiles (ISO 11898, 1993). In recent years, CAN has been used in embedded control systems that require high safety and reliability because of its appealing features and low implementation costs (Navet et al., 2005; Johansson et al.; 2005). Appealing features of CAN protocols are that the error detection mechanisms can identify multiple types of error (e.g. bits error, bit stuffing error, cyclic redundancy checksum error, frame error, and acknowledgement error). Moreover, error counters in a CAN controller can be used to represent which states of the controller are associated with specific errors, which include an error-active state, an error-passive state, and a bus-off state (Gaujal & Navet, 2005).

In spite of low implementation costs and wide acceptance of the CAN protocol in automotive control systems and industrial factory automation, limited bandwidth and nondeterministic response time have restricted the wider use of CAN in safety-critical real-time embedded control systems such as x-by-wire applications (Rushby, 2003; Wilwert et al., 2004). To mitigate the effects of these problems, the worst-case response time of a CAN message should be reduced as much as possible. Calculating the worst-case response time of CAN messages has been studied in order to guarantee its schedulability (Tindell et al., 1994, 1995), and this approach has been cited in over 200 subsequent papers. More recently, the schedulability analysis of CAN has been studied as the revised version of the original approach (Davis & Burns, 2007).

To reduce the length of CAN messages, the pre-processing mechanism using bitwise manipulation before bit stuffing has been suggested (Nolte et al., 2002, 2003). According to this mechanism, the worst-case response time can be reduced by minimizing stuffing bits in CAN messages. However, this mechanism cannot be applied to CAN network systems because the problem of message priority inversion has not been addressed. In our previous work, to resolve the problem of message priority inversion, a mechanism with a new bit mask for reducing the length of CAN message as well as preserving message priorities has been proposed (Park et al., 2007). Subsequently, we found that the mechanism has a problem which causes the frame shortening error and proposed the advanced bit stuffing (ABS) mechanism for resolving the problems with the previous approach (Park & Kang, 2009). In this paper, we describe the ABS mechanism in detail and extend the generation procedure of the bit mask of the ABS mechanism for the extended 2.0B frame format.

The outline of this paper is as follows. Section 2 presents a summary of the CAN protocol, and describes the impossibility of the worst-case bit stuffing scenario proposed by Nolte et al. Then calculating response time of CAN messages is presented. In Section 3, the ABS mechanism for reducing CAN message response time using generation of a new mask is described in detail. Also, we describe the examples of problems with priority inversion and frame shortening error. In Section 4, we evaluate the performance of the ABS mechanism with various CAN message sets. Finally, Section 5 concludes the paper.

2. Background

2.1 CAN message frame format

Controller area network (CAN) is the ISO standard for communication in automotive applications. It is designed to operate at network speeds of up to 1 Mbps for message transmission. Each CAN message contains up to 8 bytes of data (Farci et al., 1999). The frame format of a CAN message is classified into two categories that include the standard 2.0A with 11-bit identifier and the extended 2.0B with 29-bit identifier. Furthermore, message transmission over a CAN is controlled by four different types of frame: data frame, remote transmit request (RTR) frame, overload frame, and error frame (Etschberger, 2001). Fig. 1 shows the format of a CAN standard 2.0A data frame.

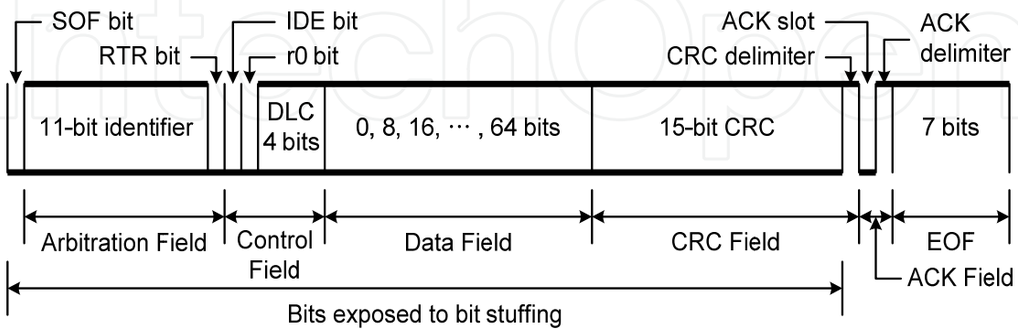

Fig. 1. Standard 2.0A data frame format of a CAN message.

As shown in fig. 1, a data frame consists of start-of-frame (SOF), arbitration field, control field, data field, acknowledgement (ACK) field, and end-of-frame (EOF). An SOF bit marks

the beginning of a data frame, and is represented by one dominant bit (value=0). The arbitration field consists of an 11-bit identifier and a dominant delimiter bit. The identifier indicates the priority of the message. A message with identifier '00000000000' has highest priority, and a message with identifier '11111111111' has lowest priority. The last four bits of the control field are called the data length code (DLC). Its value represents the length of data field. Data field contains up to 8 bytes of data to be transmitted. The CRC field consists of a 15-bit CRC code and a recessively transmitted delimiter bit. The ACK field has two delimiter bits. The EOF consists of a sequence of 7 recessive bits (Etschberger, 2001).

In recent years, a luxury car may incorporate as many as 2500 signals exchanged by up to 70 ECUs (Albert, 2004). In the standard 2.0A frame format of a CAN message, the length of the identifier is 11 bits. This means that 2048 different CAN messages are distinguishable in the CAN communication system, so, the 11-bit identifier is insufficient to distinguish all signals. For this reason, the extended 2.0B frame format with 29-bit identifier has been defined. Fig. 2 shows the format of a CAN extended 2.0B data frame (Pfeiffer et al., 2003)

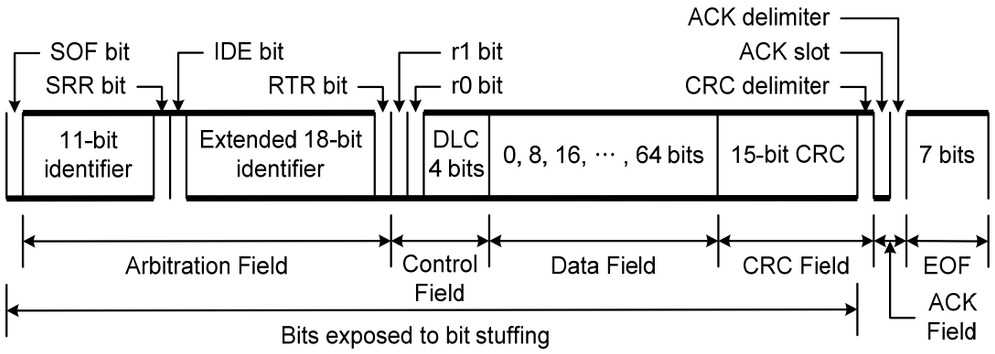

Fig. 2. Extended 2.0B data frame format of a CAN message.

2.2 Worst-case bit stuffing scenario

When a CAN node detects an error in a transmitted message, it transmits an error flag which consists of six bits of the same polarity. The bit stuffing mechanism prevents six consecutive bits from having the same polarity by inserting a bit of opposite polarity after the fifth bit. Moreover, the main purpose of the bit stuffing mechanism is used to synchronize transmitter and receiver when the same values are to be transmitted consecutively (Nolte et al., 2001). Bits exposed to bit stuffing are from an SOF bit to a 15-bit CRC code without a CRC delimiter (see Fig. 1 and Fig. 2). The stuffing bits of the received frame are removed at the receiving node before the message is processed (Wolfhard, 1997).

The worst-case scenario of the bit stuffing has been presented as shown in Fig. 3 (Nolte et al., 2007).

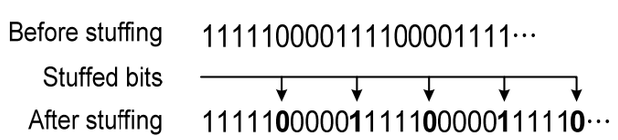

Fig. 3. The worst-case scenario of the bit stuffing.

According to the worst-case scenario, the number of bits of a CAN message is given by:

$$g + 8s + 13 + \left\lfloor \frac{g+8s-1}{4} \right\rfloor \quad (1)$$

where g is 34 for the standard format or 54 for the extended format, s is the number of data bytes of a CAN message (Nolte et al., 2007). However, it is impossible that stuffing bits be inserted in the worst-case scenario. The causes are:

1. Several bit values are fixed by the CAN frame format (e.g., the SOF bit and delimiter bits in the arbitration field and the control field).
2. The DLC in the control field depends on its number of data bytes.
3. The CRC field of a CAN message depends on the bit sequence from an SOF bit to the data field.

Accordingly, the worst-case number of stuffing bits in a standard 2.0A data frame is reduced by 21~40% from previous values.

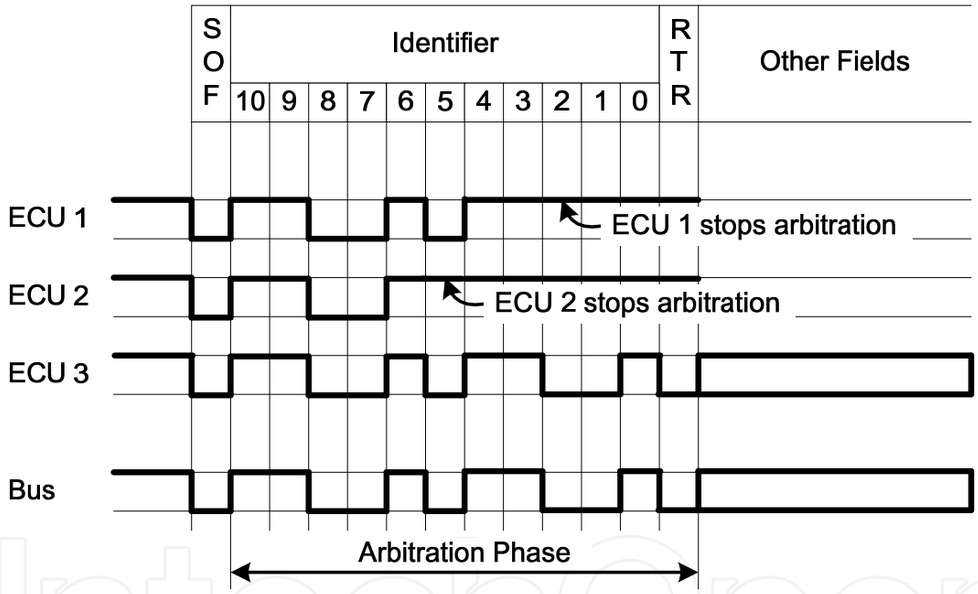

Fig. 4. The CAN arbitration process.

2.3 Bitwise bus arbitration

Each ECU of a CAN network can initiate the transmission of a message as soon as the bus is free. Because it may happen that more than one ECU begins a message transmission at the same time, an arbitration process is necessary. To prevent the ECUs from destroying each other's transmitted data, the message with the highest priority of all simultaneously arbitrating messages is determined in an arbitration phase. As mentioned in Section 2.1, the message having the lowest value message identifier is assigned highest priority (Etschberger, 2001). Fig. 4 shows the CAN arbitration process. Each ECU monitors the signal level on the bus during the arbitration phase. The arbitation phase consists of the transmission of the message identifier and of the RTR bit. If an ECU detects a dominant bus

level, although it has switched a recessive level itself, it aborts the transmission process immediately, as in this case a message with higher priority is obviously being transmitted at the same time, and goes into the receive state (Etschberger, 2001).

2.4 Response time model of CAN messages

The worst-case response time of a CAN message m can be calculated as the sum of the queuing delay t_m and the transmission delay C_m as follows:

$$R_m = t_m + C_m \tag{2}$$

The queueing delay t_m is composed of the blockng time B and the interference and is given by:

$$t_m = B + \sum_{\forall j \in hp(m)} \left\lceil \frac{t_m + J_j + \tau_{bit}}{T_j} \right\rceil C_j \tag{3}$$

where the set $hp(m)$ consists of all the messages in the system of higher priority than message m, J_j is the jitter on the queueing of the message j, T_j is the transmission period of the message j, and τ_{bit} is the transmission time for a single bit.

The blocking time B can be calculated by the transmission time of the longest CAN message within the system. The transmission delay C_m can be calculated by multiplying the number of bits of the message m as in (1) and τ_{bit} (Tindell et al., 1995).

3. Advanced Bit Stuffing (ABS) mechanism

As mentioned in Section 1, the mechanism proposed by Nolte et al. cannot be applied to reduce the length of CAN messages because the message transmission priorities can be shuffled as shown in Table 1.

	High priority CAN ID	Low priority CAN ID	Description
Original CAN Message	00001010111…	01111110101…	Low bit value has higher priority
XORing (by Nolte et al.)	0101<u>1111</u>101…	00101011111…	Mask: 01010101010
Bit Stuffing Message	010111110101…	001010111110…	Priority inversion occurs.

Table 1. Counter example of priority inversion problem.

To solve the problem of priority inversion, a mechanism for minimizing the length of CAN messages in bit stuffing, and for preserving message priorities, has been proposed (Park et al., 2007). However, the previous mechanism contains a flaw which causes frame shortening errors. The frame shortening error means that the receiver anticipates a frame of different length than the original (Charzinski, 1994; Tran, 1999). It can occur when the first bit and the last four bits of the control field are changed by the XOR operations with the bit mask. An example of the frame shortening error is shown in Fig. 5.

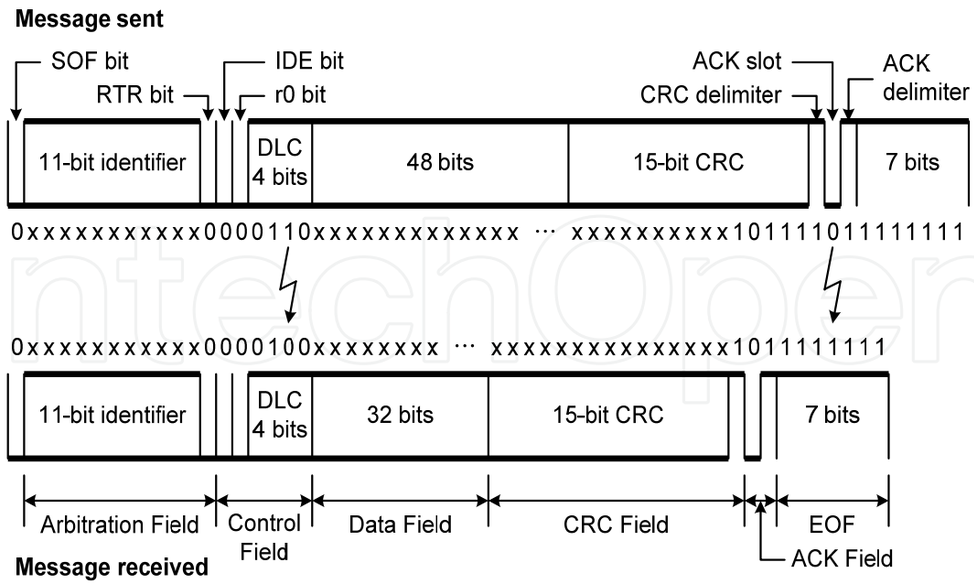

Fig. 5. Example of the frame shortening error.

As shown in Fig. 5, two bits are changed by bit errors. The first error is the bit in the DLC and the second one is the ACK slot. Thus the receiver can expect a message with a smaller than the original message.

In this section, we propose an advanced bit stuffing (ABS) mechanism which adopts XOR operations and prevents priority inversion and frame shortening errors at the same time. In order to develop the ABS mechanism, an assignment scheme for CAN message identifiers and generation rules of a new XOR bit mask are presented.

3.1 Message identifier assignment

To better understand the number of bits used for message identifiers in CAN-based control systems, we assumed that there are two assignment schemes of message identifiers. The first scheme is to assign to messages consecutive identifiers starting from 1. In this scheme, when the number of message identifiers is n, the number of used bits is $\lceil \log_2 n \rceil$. For instance, if the system requires 256 messages, than the number of used bits for message identifiers are 8. On the other hand, the second scheme is based on the grouping of message identifiers in accordance with their level of importance. In this scheme, the number of used bits can be calculated by:

$$n_{used} = \lceil \log_2 m \rceil + \lceil \log_2 n \rceil \tag{4}$$

where m and n represent the number of groups and the maximum number of identifiers in a group, respectively. For example, the system requires 4 message groups and each group consists of up to 32 message identifiers. In this case, only 7 bits are used for message identifiers. 2 bits out of 7 bits are required for representing message groups and 5 bits out of 7 bits are required for representing message identifiers of each group. Because the first scheme is a special case (i.e., $m = 1$) of the second scheme, in this paper, we have applied the second scheme in order to assign message identifiers.

3.2 XOR mask generation for standard 2.0A frame format

In order to generate a new mask for standard 2.0A frame format, we assumed that the bits for group and those for group identifiers are assigned to the most significant bit (MSB) and the least significant bit (LSB) field in the arbitration field, respectively. When the number of data bytes (s), the number of groups (m) and the maximum number of identifiers in a group (n) are determined, the following mask generation procedure is constructed for the standard 2.0A frame format.

1. The length of a mask is the length of the bits exposed to bit stuffing, $8s + 34$. The mask is initially set to "010101…"
2. A value of 0 is assigned to $\lceil \log_2 m \rceil$ bits of the MSB and $\lceil \log_2 n \rceil$ bits of the LSB of the 11-bit identifier in the mask.
3. A value of 0 is assigned to the RTR bit of the arbitration field in the mask.
4. The string "010000" is assigned to 6 bits of the control field in the mask. Then, the mask can be generated as depicted in Fig. 6.

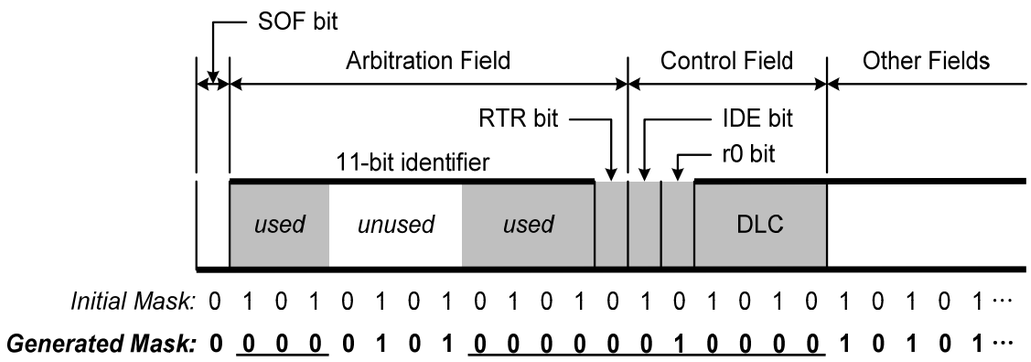

Fig. 6. XOR mask generation for standard 2.0A frame format.

3.3 XOR mask generation for extended 2.0B frame format

We extended the mask generation procedure in Section 3.2 for extended 2.0B frame format. In the same manner, we assumed that the bits for a group and those for group identifiers are assigned to the MSB and the LSB field in the arbitration field of the extended 2.0B frame format, respectively. When the number of data bytes (s), the number of groups (m) and the maximum number of identifiers in a group (n) are determined, the following mask generation procedure is constructed for the extended 2.0B frame format.

1. The length of a mask is the length of the bits exposed to bit stuffing, $8s + 54$. The mask is initially set to "010101…"
2. A value of 0 is assigned to $\lceil \log_2 m \rceil$ bits of the MSB and $\lceil \log_2 n \rceil$ bits of the LSB of the 29-bit identifier in the mask.
3. A value of 00 is assigned to 2 medial bits of the arbitration field, and a value of 0 is assigned to the RTR bit of the arbitration field in the mask.
4. The string "010000" is assigned to 6 bits of the control field in the mask. Then, the mask can be generated as depicted in Fig. 7.

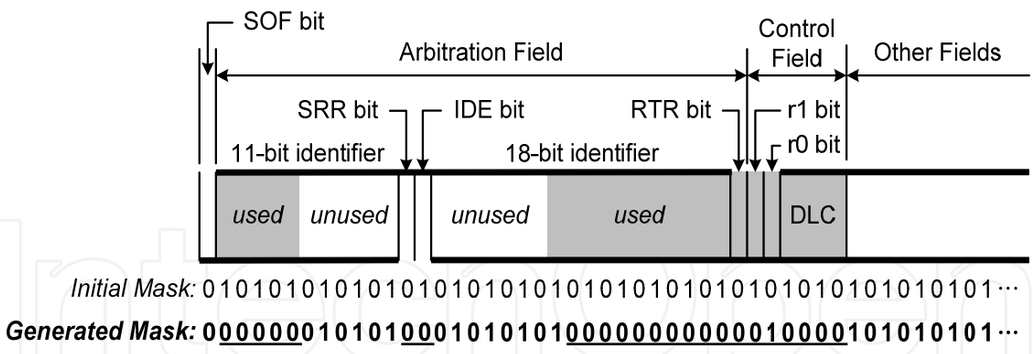

Fig. 7. XOR mask generation for extended 2.0B frame format.

In both standard and extended frame format, r0 bit should be a dominant bit. But the bit of r0 bit location in the generated mask is assigned to a recessive bit in the generation procedure for both standard and extended frame format. This assignment is for separating XOR masked CAN messages from original CAN messages. If an ECU receives a CAN message with dominant bit of r0 bit location, the received message is unmasked, and otherwise (i.e., a CAN message with recessive bit level of r0 bit location), a received message is masked.

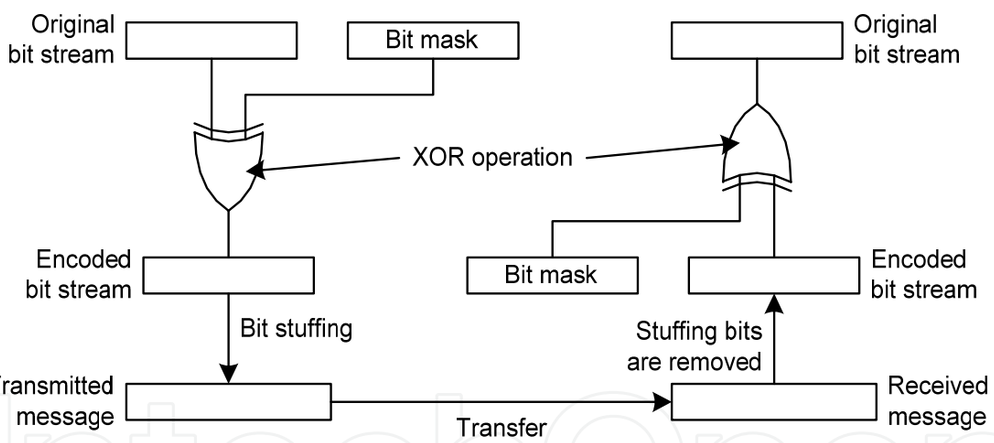

Fig. 8. Implementing the ABS mechanism

3.4 Guidance for implementing the ABS mechanism

In Section 3.2 and Section 3.3, the XOR masks are generated for standard and extended frame format. The ABS mechanism reduces the number of stuffing bits in the CAN message by a bitwise manipulation using the XOR masks (Fig. 8).

4. Performance evaluation

As in our previous work, the example of the case of $\lceil \log_2 m \rceil = 2$ illustrates the efficiency of the proposed mechanism (Park et al., 2007). For the SOF bit, the arbitration field, and the control field, the expected number of stuffing bits has been calculated with a variable

number of $\lceil \log_2 n \rceil$. The expected number of stuffing bits in the original messages is shown in Table 2. For comparison, the expected number of stuffing bits in the XOR masked messages is shown in Table 3, where $P(X = i)$ represents the probability that the message has i stuffing bits, and $E(X)$ is the expected number of the stuffing bits. In order to calculate the probability, we assume that the probability of a bit having value 1 or 0 is equal.

$\lceil \log_2 n \rceil$	$P(X = 0)$	$P(X = 1)$	$P(X = 2)$	$P(X = 3)$	$E(X)$
1	0	0.234	0.594	0.172	1.938
2	0	0.305	0.609	0.086	1.781
3	0	0.324	0.633	0.043	1.719
4	0.086	0.402	0.477	0.035	1.461
5	0.157	0.477	0.341	0.025	1.234
6	0.199	0.509	0.268	0.024	1.117
7	0.251	0.561	0.176	0.012	0.949
8	0.260	0.572	0.159	0.009	0.917
9	0.260	0.572	0.159	0.009	0.917

Table 2. The expected number of the stuffing bits in the original messages.

In order to evaluate the performance of the ABS mechanism, the number of stuffing bits of masked message frames is compared with the number of stuffing bits of the original message frames. The original and masked message frames are generated by a simulation program. The flowchart of the simulation program is shown in Fig. 9. Simulation procedures are as follows: 1) Initially, the simulation program obtains input parameters. In this simulation, the number of runs is 1000 and each CAN message has four data bytes; 2) Then, the original message frame is generated and the number of stuffing bits is calculated; 3) Next, the bit mask is generated by using mask generation as shown in Section 3.2 and Section 3.3, and masked message frame generated by XOR operation in both the bit mask and the original message frame; 4) Finally, the number of stuffing bits of the masked message frame is calculated. When a simulation is complete, the average and maximum number of stuffing bits in the original and masked message frames are calculated, respectively.

In this experiment, the number of runs is 1000 and the assumptions that are made in these experiments are as follows: 1) the probability that a bit which does not have a specific value can be set to 0 or 1 is the same, and 2) it is independent of other bit values, and finally, 3) there is no transmission error during experiment.

$\lceil \log_2 n \rceil$	$P(X=0)$	$P(X=1)$	$P(X=2)$	$P(X=3)$	$E(X)$
1	1	0	0	0	0
2	1	0	0	0	0
3	0.875	0.125	0	0	0.125
4	0.875	0.125	0	0	0.125
5	0.875	0.125	0	0	0.125
6	0.844	0.156	0	0	0.156
7	0.754	0.234	0.012	0	0.258
8	0.728	0.254	0.018	0	0.291
9	0.728	0.254	0.018	0	0.291

Table 3. The expected number of the stuffing bits in the XOR masked messages.

Fig. 10 and Fig. 11 show the average and maximum number of stuffing bits in the standard data frame as the number of used bits in the arbitration field changes, in both the original frame and the masked frame. In the same manner, the number of stuffing bits in the extended data frame is shown in Fig. 12 and Fig. 13. Here we can see that the more the number of used bits decreases, the more does the number of stuffing bits of the original message frame increase. However, the number of stuffing bits of the masked message frame showed little variation regardless of the change in the number of used bits.

From the experimental results, it can be found that the expected number of stuffing bits is reduced by the maximum of 58.3% with an average of 32.4%. This may effect the response time of CAN messages because message response time is in proportion to the length of message frame.

The average and maximum number of stuffing bits in the standard data frame with $\lceil \log_2 m \rceil = 2, \lceil \log_2 n \rceil = 4$ as the number of data bytes changes, in both the masked frame and the original frame, are shown in Fig. 14. In the same manner, the number of stuffing bits in the extended data frame is shown in Fig. 15. As shown in Fig. 14 and Fig. 15, the number of stuffing bits of the masked message frame is smaller than the number of stuffing bits of the original message frame.

As mentioned in Section 2.4, the response time of CAN messages is determined by the queuing delay and the transmission delay. If the number of stuffing bits decreases, both the queuing delay and the transmission delay can be reduced. Furthermore, this may have effects on the reduced network utilization of the embedded systems using CAN network protocols.

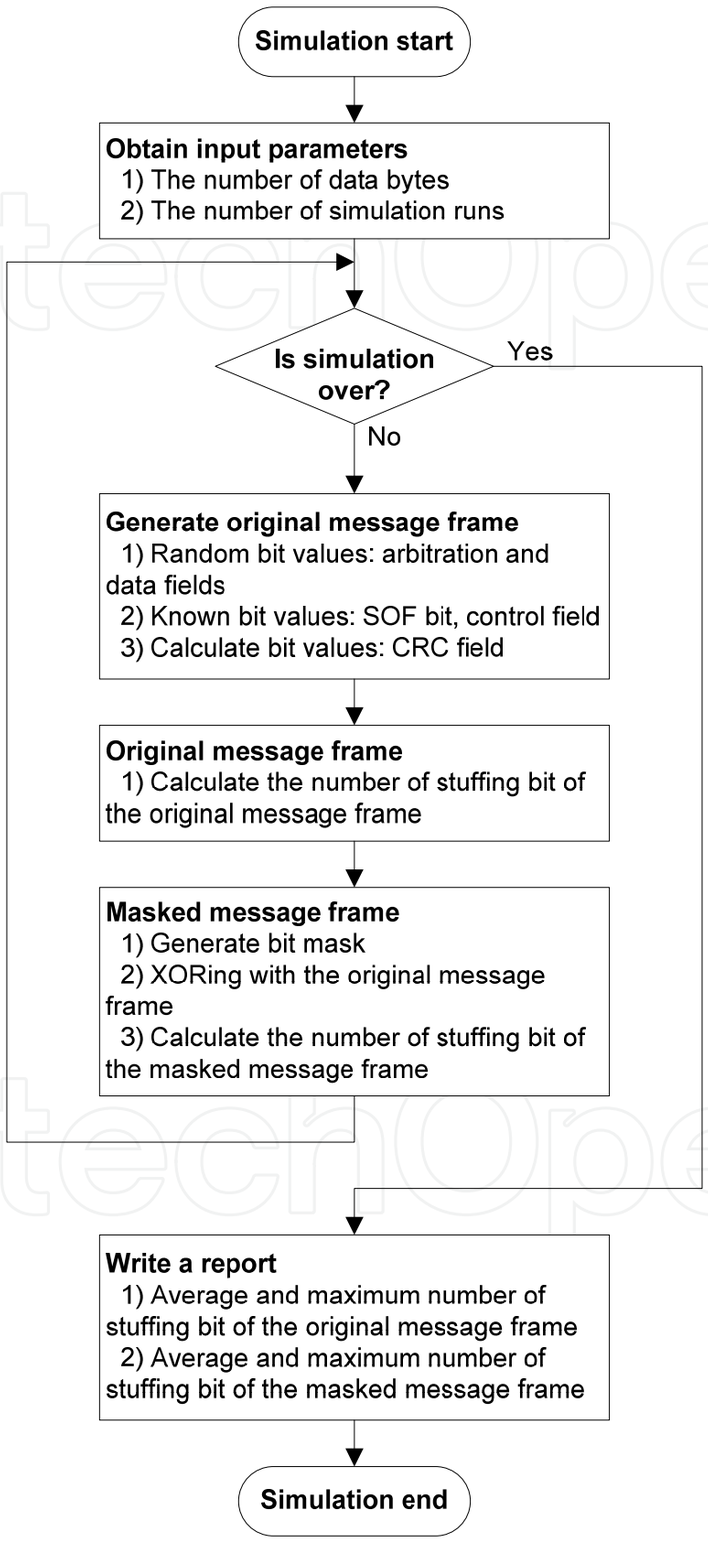

Fig. 9. Flowchart of the simulation program.

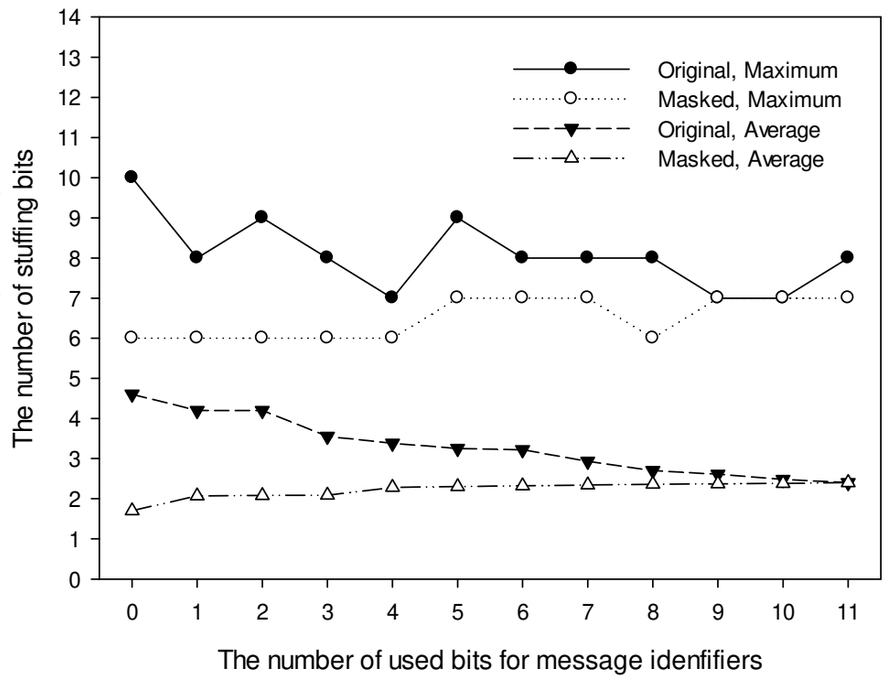

Fig. 10. The number of stuffing bits in the standard frame with 4 data bytes.

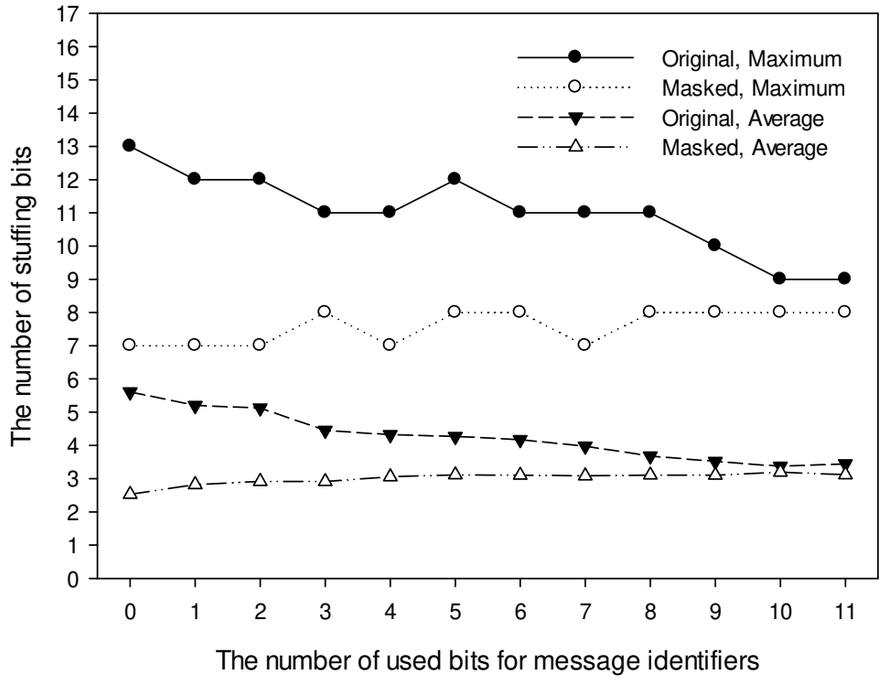

Fig. 11. The number of stuffing bits in the standard frame with 8 data bytes.

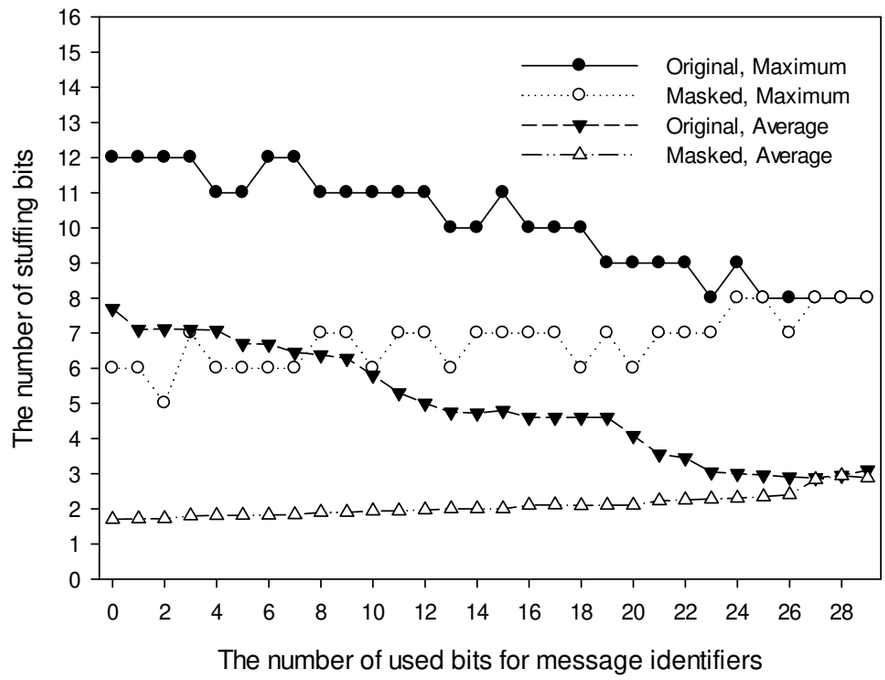

Fig. 12. The number of stuffing bits in the extended frame with 4 data bytes.

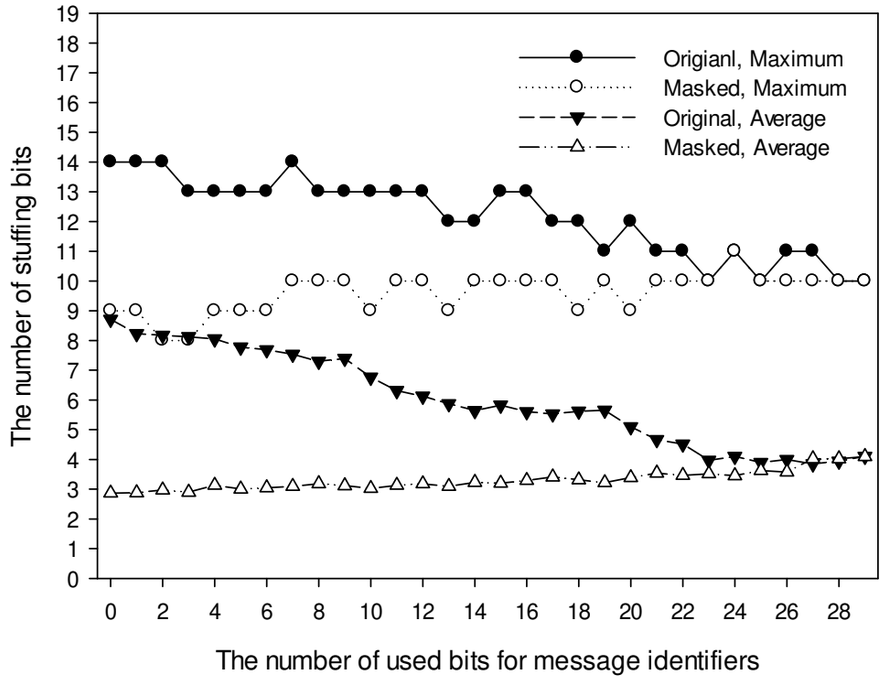

Fig. 13. The number of stuffing bits in the extended frame with 8 data bytes.

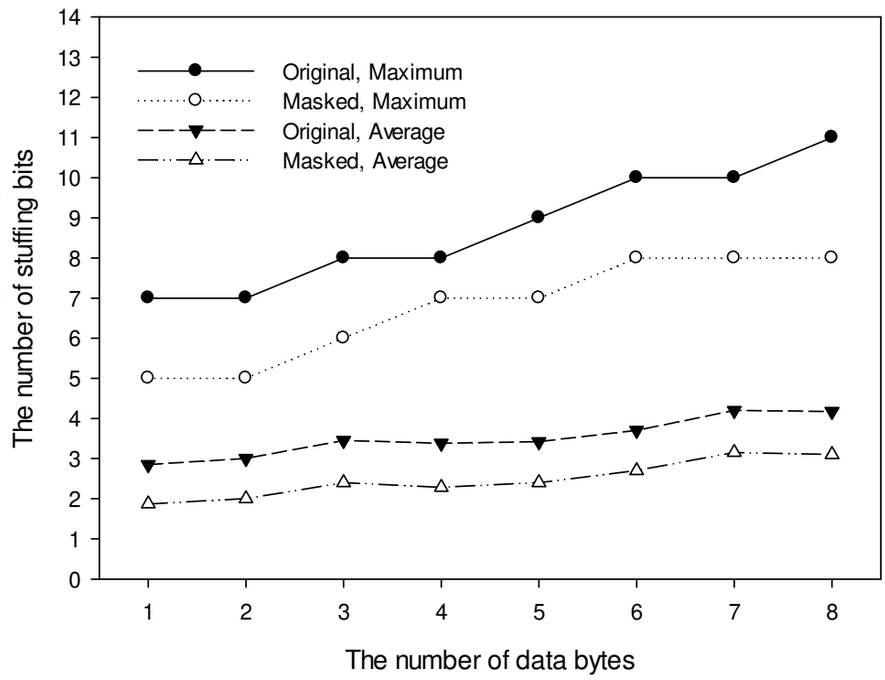

Fig. 14. The number of stuffing bits in the standard frame with $\lceil \log_2 m \rceil = 2, \lceil \log_2 n \rceil = 4$

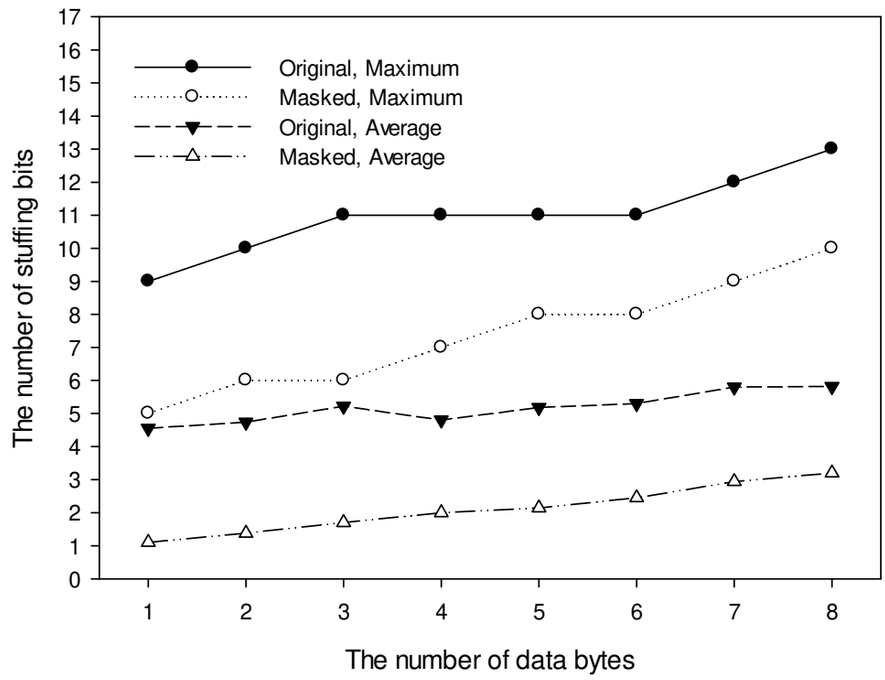

Fig. 15. The number of stuffing bits in the extend frame with $\lceil \log_2 m \rceil = 6, \lceil \log_2 n \rceil = 9$

5. Conclusions

Minimizing the response time of the CAN message is necessary to guarantee real-time performance improvement. In this paper, an effective mechanism called advanced bit stuffing (ABS) mechanism is presented. The ABS mechanism develops an assignment scheme of CAN message identifiers and generation rules of a new XOR bit mask, to prevent the problems with message priority inversion and frame shortening error. From the experimental result, the number of stuffing bits of the masked message frame showed little variation regardless of the change of the number of used bits. It has been also found that it is more effective in embedded systems in which the number of CAN messages is less than that of bits that are used for message priority.

6. References

Albert, A. (2004). Comparison of Event-triggered and Time-triggered Concepts with Regards to Distributed Control Systems, *Embedded World Conference,* Nürnberg, Germany.

Charzinski, J. (1994). Performance of the Error Detection Mechanisms in CAN, *Proceedings of the 1st International CAN Conference,* Mainz, Germany.

Davis, R. & Burns, A. (2007). Controller Area Network (CAN) Schedulability Analysis: Refuted, Revisited and Revised, *Real-Time Systems,* Vol. 35, No. 3, pp. 239-272.

Etschberger, K. (2001). *Controller Area Network (CAN): Basics, Protocols, Chips, and Applications*, IXXAT Automation GmbH, ISBN 978-3000073762, Weingarten, Germany.

Fabian, S. & Wolfgang, S. (2006). Time-Triggered vs. Event-Triggered: A Matter of Configuration, *Proceedings of the GI/ITG Workshop on Non-Functional Properties on Embedded Systems,* pp. 1-6.

Farsi, M.; Ratcliff, K.; & Barbosa, M. (1999). An Overview of Controller Area Network, *Computing & Control Engineering Journal,* Vol. 10, pp. 113-120.

Gaujal, B. & Navet, N. (2005) Fault Confinement Mechanisms on CAN: Analysis and Improvements, *IEEE Transactions on Vehicular Technology,* Vol. 54, pp. 1103-1113.

International Standards Organization. (1993). Road Vehicles - Interchange of Digital Information – Controller Area Network (CAN) for High-Speed Communication. ISO 11898.

Johansson, K.H.; Torngren, M.; & Nielsen, L. (2005). Vehicle Applications of Controller Area Network, *Handbook of Networked and Embedded Control Systems,* ISBN 0-8176-3239-5, pp. 741-766.

Navet, N.; Song, Y.; Simonot-Lion, F.; & Wilwert, C. (2005). Trends in Automotive Communication Systems, *In Proceeding of the IEEE,* Vol. 93, No.6, pp. 1204-1223.

Nolte, T.; Hansson, H.; Norstrom, C.; & Punnekkat, S. (2001). Using bit-stuffing distributions in CAN analysis, *IEEE/IEE Real-Time Embedded Systems Workshop (RTES'01).*

Nolte, T.; Hansson, H.; & Norstrom, C. (2002). Minimizing CAN Response-Time Jitter by Message Manipulation, *Eighth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02),* pp. 197-206.

Nolte, T.; Hansson, H.; & Norstrom, C. (2003). Probabilistic Worst-Case Response Time Analysis for the Controller Area Network, *Ninth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'03),* pp. 200-207.

Obermaisser, R. (2004). *Event-Triggered and Time-Triggered Control Paradigms,* Sptinger-Verlag, ISBN 978-1441935694.

Park, K.; Kang, M.; & Shin, D. (2007). Mechanism for Minimizing Stuffing-bits in CAN Messages, *The 33rd Annual Conference on the Industrial Electronics Society (IECON '07),* pp. 735-737.

Park, K. & Kang, M. (2009). Advanced Bit Stuffing Mechanism for Reducing the Length of CAN Messages, *International Conference on Convergence Technologies and Information Convergence (CTIC'09).*

Pfeiffer, O.; Ayre, A.; & Keydel, C. (2003). *Embedded Networking with CAN and CANopen,* RTC Books, ISBN 978-0929392783.

Rushby, J. (2003). A Comparison of Bus Architecture for Safety-Critical Embedded Systems, *NASA/CR, Technical Report,* NASA/CR-2003-212161.

Tindell, K.W. & Burns, A.K. (1994). Guaranteed Message Latencies for Distributed Safety-critical Hard Real-time Networks, *Technical Report YCS 229,* Department of Computer Science, University of York.

Tindell, K.; Burns, A.; & Wellings, A. J. (1995). Calculating Controller Area Network (CAN) Message Response Times, *Control Engineering Practice,* Vol. 3, No. 8, pp. 1163– 1169.

Tran, E. (1999). *Multi-bit Error Vulnerabilities in the Controller Area Network Protocol,* Carnegie Mellon University, Thesis.

Wilwert, C.; Navet, N.; Song, Y.-Q.; & Simonot-Lion, F. (2004). Design of automotive X-by-Wire systems, *The Industrial Communication Technology Handbook,* R. Zurawski, Ed. Boca Raton, FL: CRC.

Wolfhard, L. (1997). *CAN System Engineering: From Theory to Practical Applications,* Springer, ISBN 978-0387949390.

IntechOpen