A Framework for Fragmenting/Reconstituting Data Frame in Controller Area Network (CAN)

Changmin Shin*

*Electronics and Telecommunications Research Institute, South Korea

cmshin@etri.re.kr

Abstract— **Controller Area Network (CAN) is a multi-master, message-based serial network communication protocol. Since the "Data field" of the CAN Frame supports the size of 8 bytes, the CAN transmitting node cannot transfer data beyond the size of 8 bytes. The purpose of this paper is to provide a framework for the transmission of long data beyond the size of 8 bytes in CAN network. The framework consists of transmitter node and receiver node. The transmitter node splits the long data into several small messages. And the receiver node reconstitutes the split messages into the original data. This paper proposes the detailed process of two nodes using a new defined CAN Frame format proposed in the previous paper. In the previous work, there isn't the process for fragmenting/reconstituting CAN data frame.**

Keywords— **Controller Area Network (CAN), Data Fragmentation, Data Synthesis**

I. INTRODUCTION

Currently, there are many mechanical devices and a variety of Electronic Control Unit (ECU) to control the devices inside the car. The many advanced electronic devices are mounted at the future car to provide driving comfort, safe driving [1].

In addition, as services such as automated equilibrium control, the smart cruiser control, and self-parking will be expanded gradually, electronic devices are connected to each other. Accordingly, the importance of the vehicle automotive embedded software platform and the in-vehicle communication (In-Vehicle Network) is required.

As the number of electronic devices to be applied to the vehicle is increased gradually, the importance of In-Vehicle Network increases as well. Depending on the transfer rate and the attributes of the data, the various standards (LIN, FlexRay, CAN, MOST) for In-Vehicle Networks exists.

FlexRay is a hybrid protocol that has both advantages of Event-triggered and Time-triggered paradigm, and has emerged as the de-facto standard for in-vehicle automotive communication. FlexRay was developed through a consortium founded by a BMW, Bosch, DaimlerChrysler, Philips, etc. [2,3,4].

CAN (Controller Area Network) was developed by BOSCH for application in the automotive industry. CAN is now widely applied in various industries in addition to the automotive industry. CAN protocol is a multi-master, message-based serial network communication protocol that provides a maximum signalling rate of 1 Mbps defined in the ISO-11898 standard [5,6].

CAN supports communications between different Electronic Control Units (ECUs) in an automobile, which are connected through twisted-pair cables supporting half-duplex protocol and a Carrier Sense Multiple Access/Collision Detection with Arbitration on Message Priority (CSMA/CD+AMP) protocol. CAN secures high reliability such as high noise immunity, error detection, and error collection.

Because the data field of a CAN data frame has a maximum size of 8 bytes, a CAN transmitter node may not transmit data with a size of more than 8 bytes.

The purpose of this paper is that CAN transmitting node supports the transmission of long messages beyond the size of 8 bytes. Thus, this paper proposes a framework for the transmission of long messages. The framework uses a new CAN Frame format in the previous paper and the new defined CAN Frame has a mutually compatible with the existing CAN Frame.

The summarized processes for transferring long data are as follows.

First, this paper adopted a new defined CAN Frame format proposed in the previous paper.

Second, if the data size to be transmitted is more than 8 bytes, the transmitting node splits the data in six bytes. The sending node transmits the split messages multiple times in order. The adopted CAN Frame format is used in the split messages.

Finally, the receiving node synthesizes the split messages sent from the sending node. The split messages include important information for reconstituting the original data frame.

If the data size to be transmitted does not exceed 8 bytes, the existing CAN protocol method is applied.

The rest of this paper is organized as follows. Section 2 describes a new defined CAN Frame structure for the transmission of long messages. Section 3 presents the framework for the transmission of long messages proposed in the paper. Especially, section 3 depicts the detailed process of fragmenting/reconstituting CAN frame. Section 4 presents conclusion in proposed method.

II. TRANSMISSION METHOD OF LONG MESSAGES

The transmission of long messages needs a new CAN 2.0 frame structure. So, this section depicts the structure of a CAN 2.0 data frame and the new defined structure of a CAN 2.0 data frame proposed in the previous paper [1].

A. CAN 2.0 Frame

Figure 1 is a diagram illustrating the structure of a CAN 2.0A data frame used in a CAN data frame transmitting method. Figure 2 is a diagram illustrating the structure of a CAN 2.0B data frame used in a CAN data frame transmitting method.

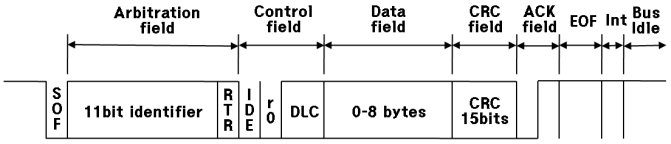

Figure 1. CAN 2.0A Frame structure

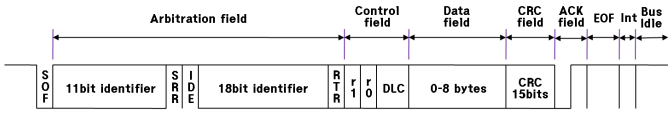

Figure 2. CAN 2.0B Frame structure

"SOF (Start Of Frame)" means the beginning of the CAN Frame. "Arbitration field" is composed of a 29-bit or 11-bit identifier and "RTR (Remote Transmission Request)". If "RTR" is "0", the message means the data Frame. If "RTR" is "1", it means to request data from other nodes. "r0" and "r1" of "Control Field" is the size of one bit, which is an area that is reserved for future use. "DLC (Data Length Code)" which is a size of 4 bits indicates the size of the "Data field". The "Data field" can include a data of 0 to 8 bytes. And, "EOF (End Of Frame)" means the end of the Frame.

B. CAN 2.0 Frame for long messages

Figure 3 depicts a new CAN 2.0A Frame structure proposed in this paper. Figure 4 depicts a new CAN 2.0B Frame structure proposed in this paper. The defined CAN 2.0A Frame structure is the same as the CAN 2.0A Frame structure excepting two differences. The two differences are the use of "r0" and the reconstituted "Data field". In this way, the proposed format has used only the "Data field" and "r0" of traditional CAN Frame structure. So, it is possible to maintain compatibility with existing CAN frame.

"r0" is used to Fragmentation Flag. If the size of the data that transmission node transmits is longer than 8 bytes, "r0" is set to "dominant". Otherwise, "r0" is set to "recessive".

"Data field" of the proposed CAN Frame can be divided into four areas such as "Message Identifier", "Seq. Number", "Seq. Delimiter" and, "Data".

"Message Identifier" is an identifier of the CAN message. And, "Seq. Number" is used to distinguish the order of the split messages. "Seq. Delimiter" is set to the "recessive" when the last message of the split messages is transmitted. Otherwise, "Seq. Delimiter" is set to "dominant". The one of the split messages is mounted on the "Data".

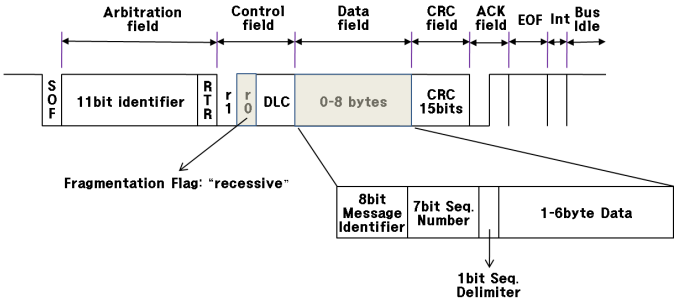

Figure 3. CAN 2.0A Frame structure for long messages

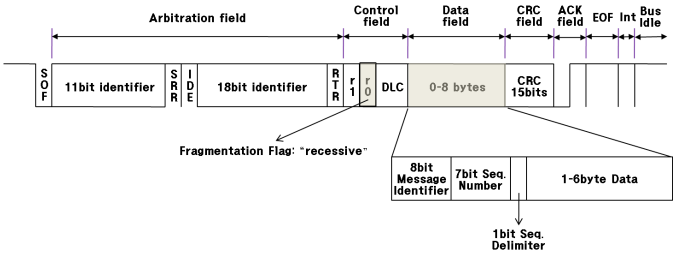

Figure 4. CAN 2.0B Frame structure for long messages

If the data size to be transmitted is more than 8 bytes, the transmitting node splits the data in six bytes. The transmitter node transmits the split messages multiple times in order. The receiver node synthesizes the split messages sent from the sending node. If the data size to be transmitted does not exceed 8 bytes, the existing CAN protocol method is applied.

III. FRAGMENTING/RECONSTITUTING CAN FRAME

One CAN message data and another CAN message data may be discriminated through a Message Identifier. For example, if both a message A and a message B are all 24 bytes long, each of the messages A and B may be fragmented to a 6-byte size to generate 4 data fragments.

When the message A is fragmented into data fragments A1, A2, A3 and A4 and the message B is fragmented into data fragments B1, B2, B3 and B4, the data fragments A1 and B1 cannot be discriminated from each other without message identifier in the data field. The message identifier may have a unique message ID that is given to each message to discriminate it from all other messages.

Thus, the data fragments A1 to A4 have the same message ID value and also the data fragments B1 to B4 have the same message ID value. However, the message ID value of the data fragments A1 to A4 is different from the message ID value of the data fragments B1 to B4.

When there are several data fragments received by a receiver node, the unique message ID of the message identifier is important for extracting data fragments from the same CAN message data and reconstituting the data fragments in accordance with the fragmentation sequence.

The Seq. Number is a place to store a sequence parameter (Seq). For example, if CAN message data exceeding 8 bytes are fragmented to a certain size (e.g., 6 bytes) prior to transmission from a transmitter node, the sequence number is necessary for a receiver node to reconstitute the received data

fragments in accordance with the data fragmentation sequence to restore the original transmitter data of the transmitter node. Thus, if the data fragmentation sequence is not predefined, it is difficult for the receiver node to restore the original transmitter data by data reconstitution.

The sequence parameter value is initialized when data fragments constitute a new CAN data frame. The sequence parameter value is initialized to '0'.

The Seq. Delimiter is a data delimitation parameter. If it is the last data fragmentation operation, the data delimitation parameter is set to 'recessive'; if not, the data delimitation parameter is set to 'dominant'.

A. CAN Frame Fragmentation

Figure 5 is a flow chart illustrating a data fragmentation process of a transmitter node.

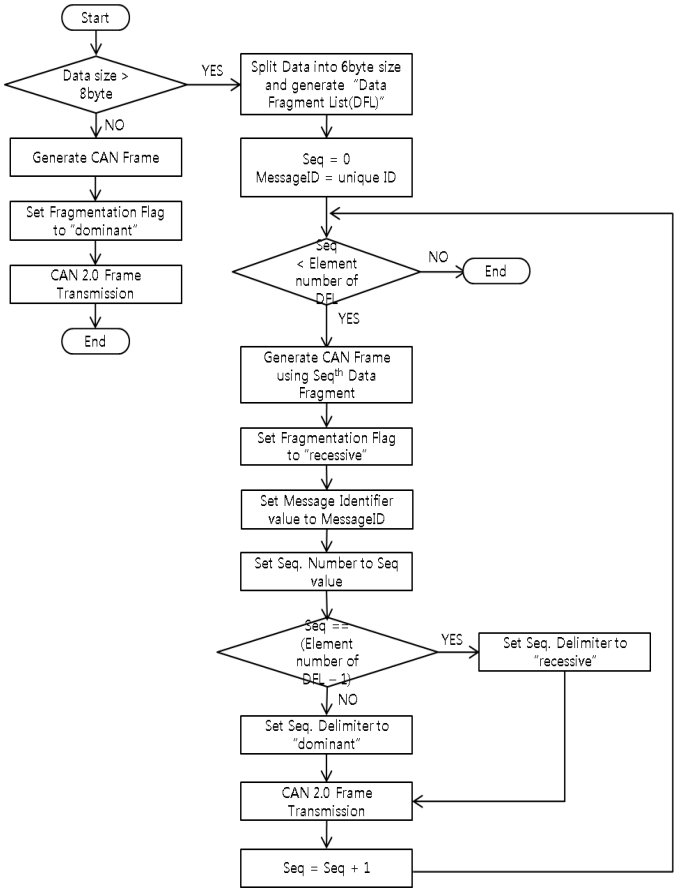

Figure 5. Data fragmentation process of a transmitter node

'Recessive' and 'dominant' bit (i.e., bit r0 or r1 of the control field) is used to indicate the data status in a CAN data frame, i.e., whether the data are fragmented data. A bit r0 or r1 may be used as a fragmentation flag. If the size of data to be transmitted by the transmitter node (hereinafter referred to as transmitter data) is larger than 8 bytes, the bit r0 or r1 is set to a recessive bit, which is logical 1; if not, it is set to a dominant bit, which is logical 0.

Referring to Figure 5, the transmitter node measures the size of transmitter data. If the transmitter data size does not exceed 8 bytes, the transmitter node generates and transmits a CAN data frame in accordance with the existing CAN protocol. In this case, bit r0 or r1 of a control field acts as a fragmentation flag, which is set to dominant.

If the transmitter data size exceeds 8 bytes, the transmitter node fragments the transmitter data.

First, the transmitter node fragments the transmitter data to a certain size (e.g., 6 bytes) and generates a Data Fragment List (DFL) in step S101. Then, the transmitter node sets a sequence parameter "Seq" to '0' and sets a message ID to a unique value different from those of other messages in step S102.

Herein, the transmitter data are fragmented to a certain size (e.g., 6 bytes) sequentially from the front of the message, and the DFL (Data Fragment List) is generated in accordance with the fragmentation sequence. In setting the sequence parameter, the data fragmentation sequence starts not from '1' but from '0'. Also, the sequence parameter is equal to the data fragment generation sequence.

If the transmitter data size is 8 bytes, the transmitter data may be transmitted without the data fragmentation operation. Herein, the sequence parameter has a value of '0', and the DFL has only one element.

If the sequence parameter value is smaller than the number of the elements of the DFL (e.g., if the number of the elements of the DFL is '2' and the sequence parameter value is '0'), the transmitter node generates a CAN data frame by using the Seq^{th} data fragment of the DFL.

Then, the transmitter node sets the fragmentation flag value of the control field in the CAN data frame to 'recessive', sets the message identifier value to the message ID, and sets the Seq. Number value to the sequence parameter value.

If the sequence parameter value is smaller by '1' than the number of the elements of the DFL, it means that the corresponding data fragment is the last data fragment. In this case, the transmitter node sets a data delimitation parameter to 'recessive'. If the corresponding data fragment is not the last data fragment, the transmitter node sets the data delimitation parameter to 'dominant'.

Thereafter, the transmitter node transmits the CAN data frame, and increases the sequence parameter value by a factor of '1'. These operations are repeated until the sequence parameter value is equal to the number of the elements of the DFL. If the sequence parameter value is equal to the number of the elements of the DFL, the data fragmentation process is ended.

B. CAN Frame Reconstitution

Figure 6 is a flow chart illustrating a data reconstitution process of a receiver node.

Referring to Figure 6, the receiver node receives a CAN data frame, and checks a fragmentation flag value. If the fragmentation flag value is not 'recessive', the receiver node extracts data from a data field and ends the data reconstitution process.

If the fragmentation flag value is 'recessive', it means that the data of the received CAN data frame are fragmented data. In this case, the following operations are performed.

First, the receiver node generates a CAN data structure including Message ID, Data Seq, and Data Fragment fields. That is, the CAN data structure includes a region for receiving the data fragment, a region for receiving a message ID value for identifying CAN message data including the data fragments, and a region for receiving a sequence parameter value containing the fragmentation sequence information.

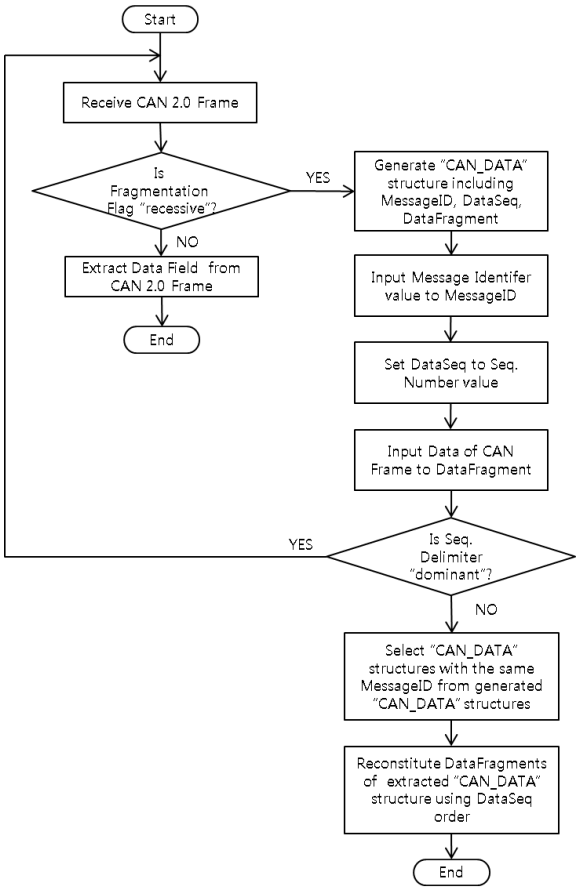

Figure 6. Data reconstitution process of a receiver node

To this end, the receiver node inputs a message identifier value to the Message ID, a sequence parameter value to the Data Seq, and the data of the CAN data frame to the Data Fragment.

Thereafter, if a data delimitation parameter is 'dominant', the receiver node waits to receive another CAN data frame to repeat the above operations.

If the data delimitation parameter is 'recessive', it is determined that the data fragment in the received CAN data frame is the last data fragment. If the data delimitation parameter is 'recessive', the receiver node extracts CAN data structures with the same message ID value from the generated CAN data structures, and reconstitutes the data fragments of the extracted CAN data structures in accordance with the sequence according to the sequence parameter value.

Thereafter, the receiver node restores the original transmitter data from the transmitter node, and ends the data reconstitution process.

IV. CONCLUSIONS

Because "Data field" of the CAN Frame supports the size of 8 bytes, there is a problem that the CAN transmitting node cannot transmit more than 8 bytes of data. The purpose of this paper is to provide a framework for the transmission of long messages.

In the proposed framework, the transmitter node splits the long data in six bytes using a new defined CAN frame proposed in the previous paper and transmits the split messages multiple times in order. The receiving node synthesizes the split messages sent from the transmitter node.

Therefore, the proposed framework can transmit more than 8 bytes of data in size.

ACKNOWLEDGMENT

This work was supported by the Technology Innovation Program (No. 10044313) funded By the Ministry of Trade, industry & Energy (MI, Korea).

REFERENCES

[1] Chang-Min Shin, "Transmission method of long messages in CAN Network", International Symposium on Embedded Technology, pp. 151-152, May 23-24, 2013.
[2] FlexRay Consortium, http://www.flexray.com/
[3] FlexRay Requirements Specification Version 2.1, http://www.flexray.com/
[4] FlexRay Communications System Protocol Specification Version 2.1 Revision A, http://www.flexray.com/
[5] CAN Specification 2.0, Robert Bosch GmbH, 1991.
[6] K. Tindell, H. Hanrson and A. Wellings, "Analysing real-time communications: controller area network (CAN)", Real-Time Systems Symposium, pp.259-263, 1994.

Changmin Shin received his B.S. degree in computer science and engineering from Korea University, Seoul, Korea in 1996. He also received an M.S. degree from Korea University in 1998. Currently, he is a Ph.D. candidate in computer science and engineering from Korea University, Seoul, Korea and is working as a senior researcher with Electronics and Telecommunications Research Institute (ETRI). His research interest is wireless multi-hop network protocols and embedded system.