Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Efficient controller area network data compression for automobile applications*#

Yu-jing WU, Jin-Gyun CHUNG‡

(*Division of Electronics & Information Engineering, Chonbuk National University, Jeonju 561-756, Korea*)

E-mail: yjwu@jbnu.ac.kr; jgchung@jbnu.ac.kr

Received Apr. 15, 2014; Revision accepted Nov. 4, 2014; Crosschecked Dec. 17, 2014

Abstract: Controller area networks (CANs) have been designed for multiplexing communication between electronic control units (ECUs) in vehicles and many high-level industrial control applications. When a CAN bus is overloaded by a large number of ECUs connected to it, both the waiting time and the error probability of the data transmission are increased. Thus, it is desirable to reduce the CAN frame length, since the duration of data transmission is proportional to the frame length. In this paper, we present a CAN message compression method to reduce the CAN frame length. Experimental results indicate that CAN transmission data can be compressed by up to 81.06% with the proposed method. By using an embedded test board, we show that 64-bit engine management system (EMS) CAN data compression can be performed within 0.16 ms; consequently, the proposed algorithm can be successfully used in automobile applications.

Key words: Controller area network (CAN), Electronic control units (ECUs), Data compression, Signal rearrangement
doi:10.1631/FITEE.1400136 **Document code:** A **CLC number:** TP274

1 Introduction

A controller area network (CAN) is a serial communication protocol which efficiently supports distributed real-time control with a very high level of security, and was first developed early in the 1980s (Bosch, 1991). The serial bus system has been successfully applied in many fields due to its high reliability and cost efficiency. The CAN system has gained popularity in embedded machine control applications such as home appliances, industrial machines, and medical equipment, which require serial communication between microcontrollers (Leen and Heffernan, 2002; Desai *et al.*, 2013).

The CAN protocol is based on a bus topology, and only two wires are needed for communication over a CAN bus. The bus has a multi-master structure where each device on the bus can send or receive data. Only one device at a time can send data while all the others listen. If two or more devices attempt to send data at the same time, the one with the highest priority is allowed to send its data while the others return to receive mode (ISO, 2003).

The majority of product innovations in the automotive industry are delivered through the increasing use of electronic control units (ECUs). There can be more than 50 ECUs distributed in a modern premium vehicle and they account for about 40% of the total value of the vehicle (Ortega *et al.*, 2006). As the number of ECUs or sensors connected to the CAN bus increases, so does the bus load. When a CAN bus is overloaded, it is not easy to transmit low-priority CAN messages due to the increased waiting time. The

‡ Corresponding author
* Project supported by the Information Technology R&D Program of MOTIE/KEIT (No. 10044092) and Research Funds of Chonbuk National University in 2013
# A preliminary version was presented at the IEEE International Symposium on Circuits and Systems, June 1–5, 2014, Australia
ORCID: Yu-jing WU, http://orcid.org/0000-0002-9339-0866; Jin-Gyun CHUNG, http://orcid.org/0000-0002-7127-4944

error probability of the data transmission also increases.

If the bus overload causes critical problems in a CAN system, the setting up of another CAN network is required. Alternatively, the bus overload can be efficiently reduced by applying a data compression technique to the CAN data. To reduce the bus load, only the differences between the current and the preceding CAN messages can be transmitted, based on the observation that CAN data (e.g., fuel or key on status data) do not change rapidly (Lawrenz, 1997).

In the adaptive data reduction (ADR) algorithm, if the value of a delta (the difference between the current and the preceding CAN signals) of two consequent signals exceeds the length of the assigned delta field, the current CAN signal is transmitted rather than the delta compressed version of the message (Ramteke and Mahmud, 2005). The ADR algorithm uses two message IDs to send CAN messages, the original message ID and the compressed message ID (which is smaller than the original message ID).

The improved adaptive data reduction (IADR) algorithm uses a single message ID to send both compressed and uncompressed messages (Miucic and Mahmud, 2006). In this algorithm, the first bit of the data field is set to '1' when a message is compressed, and set to '0' when the message is not compressed. Using the IADR algorithm, CAN signals can be fully compressed, delta compressed, or uncompressed, depending on the delta values.

By using data length code (DLC) in CAN data frame format, the enhanced data reduction (EDR) algorithm (Miucic *et al.*, 2009) eliminates the difficulties in the identification of compressed messages, such as the use of the reserved bit (Misbahuddin *et al.*, 2001), the use of dedicated message IDs (Ramteke and Mahmud, 2005), or the use of additional bits in the data field (Miucic and Mahmud, 2006). In addition, the EDR algorithm uses a method for managing signals of shorter lengths (i.e., <5 bits) by combining them into groups that are handled as single signals.

Using the boundary of fifteen compression (BFC) algorithm, if the current value of a CAN signal has changed within the maximum compression range of ±15, then the CAN signal can be compressed (Kelkar and Kamal, 2014).

In the compression area selection algorithm (Wu *et al.*, 2014), it is not necessary to predict the maximum value of the difference in successive CAN messages. In addition, the 64-bit data field is always assumed to be composed of eight signals, each with eight bits. This mechanism eliminates the need for the engineering standard of signal bit length. However, the length of the data field is additionally increased by up to 8 bits due to the header bits.

In this paper, a signal rearrangement algorithm (SRA) is proposed to obtain more compression efficiency by using signal characteristics. CAN signals are rearranged within the data field of a CAN frame based on simulation results of the actual CAN signals, such that the length of the reduced data field is minimized. In addition, as opposed to the compression area selection method, the 64-bit data field is assumed to be composed of three signals with 24, 24, and 16 bits, respectively. Thus, the number of header bits is limited to up to three bits, regardless of the number of signals in the data field of a CAN frame.

2 Existing CAN message compression methods

In this section, after the CAN data frame format is briefly introduced, three existing CAN message compression methods are reviewed.

2.1 CAN frame format

A CAN is a serial communication protocol suited for networking sensors, actuators, and other nodes in real-time systems. There are two versions of the CAN protocol: CAN 2.0A (standard CAN) with 11-bit identifiers, and CAN 2.0B (extended CAN) with 29-bit identifiers. For in-vehicle communications, only CAN 2.0A is used since it provides a sufficient number of identifiers (ISO, 2003). Fig. 1 shows the format of the CAN 2.0A data frame.

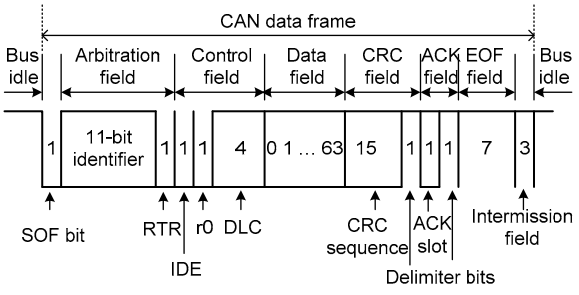

Fig. 1 Format of the CAN 2.0A data frame

A data frame begins with a start-of-frame (SOF) bit which is followed by an 11-bit identifier. The identifier and the remote-transmission-request (RTR) bit form the arbitration field. The CAN data field can contain up to eight bytes of data. The actual size of the data field is denoted by the DLC in the control field. CAN systems use non-return-to-zero (NRZ) bit representation with a bit stuffing length of five. The overall size of a CAN frame is, at most, 135 bits, including all of the protocol overheads such as the stuff bits. A CAN data field can be represented as a 2D memory map (Table 1).

Table 1 Memory map of a 64-bit CAN data field

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	7	6	5	4	3	2	1	0
Byte 1	15	14	13	12	11	10	9	8
Byte 2	23	22	21	20	19	18	17	16
Byte 3	31	30	29	28	27	26	25	24
Byte 4	39	38	37	36	35	34	33	32
Byte 5	47	46	45	44	43	42	41	40
Byte 6	55	54	53	52	51	50	49	48
Byte 7	63	62	61	60	59	58	57	56

2.2 Enhanced data reduction algorithm

By using the DLC in CAN data frame format, the EDR algorithm eliminates difficulties in the identification of compressed messages. The EDR algorithm uses a method for managing signals of shorter lengths (i.e., <5 bits) by combining them into groups that are handled as single signals.

In the EDR algorithm, the encoded message can have two types of signals. Any CAN signal with a bit length of more than five bits is the SDN (signal, delta, no-change) type. An SDN signal allows a signal to be represented in its entirety, as a delta change, or as no change. Any CAN signal with a bit length of fewer than five bits is the SN (signal, no-change) type. An SN signal allows a signal to be represented in its entirety or as no change.

In the EDR algorithm, the first byte in a compressed CAN message is the data compression code (DCC). The position of each bit in the DCC corresponds to the data byte position of the originally intended uncompressed message. A DCC bit with a value of '0' indicates that the corresponding signal is uncompressed, and a DCC bit with a value of '1' indicates that the corresponding signal has been delta compressed or fully compressed. The reduction type (RT) bit indicates the delta compressed type (RT=0) or fully compressed type (RT=1).

Table 2 shows an example of a CAN signal. Fig. 2 shows SDN and SN type groupings of the signals in Table 2. Note that this message has three SDN type signals and two SN type signals. Fig. 3 shows the compressed message obtained by the EDR algorithm. In this example, a 5-bit DCC and two RT bits are used. The EDR algorithm can achieve a data compression rate of 16.7%, since five bytes (35 bits) are required instead of the original six bytes (48 bits).

Table 2 CAN signal example

Parameter	Bit length (bit)	Previous value	Current value	Difference in value
Vehicle speed	16	21	35	14
Coolant temp	8	16	15	−1
Engine RPM	8	30	30	0
Fan RPM	4	3	6	3
Oil sensor	4	5	3	−2
Engine state	3	2	2	0
Gear state	3	1	1	0
Park state	2	1	1	0

Coolant temp: coolant temperature

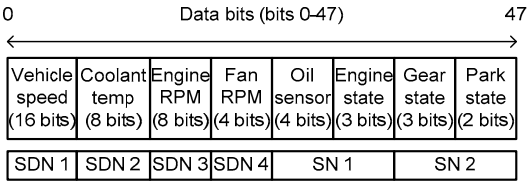

Fig. 2 SDN and SN type groupings of the signals in Table 2

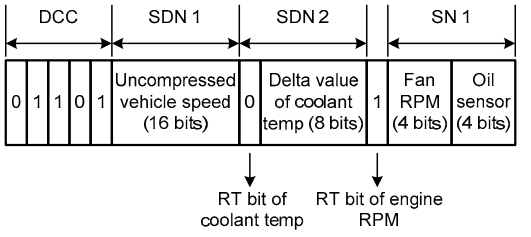

Fig. 3 Compressed message of the signal in Table 2 obtained by the EDR algorithm

2.3 Boundary of fifteen compression algorithm

If the current value of a CAN signal has changed within the maximum compression range of ±15, then

the CAN signal can be compressed using the BFC algorithm.

CAN signals of five or fewer bits are included in the class of non-boundary of fifteen (NBF) signals. CAN signals of six or more bits are included in the class of boundary of fifteen (BF) signals. Each NBF signal is allocated one bit called the bit parameter compression (BPC) bit. A BPC bit of '0' indicates that the corresponding NBF signal is in the uncompressed form. A BPC bit of '1' indicates that the NBF signal is fully compressed.

The BFC algorithm allocates parameter compression (PC) bits to every BF signal. PC bits can indicate one of four different compression scenarios (Table 3). Assume that we try to send the signals in Table 2. Vehicle speed, coolant temperature, and engine RPM are BF signals and the others are NBF signals. Based on the difference values {14, −1, 0, 3, −2, 0, 0, 0}, PC and BPC bits are determined as shown in Table 4. Fig. 4 shows the compressed data of the example in Table 2 obtained using the BFC algorithm.

In this example, the BFC algorithm can achieve a data compression rate of 33.3%, since four bytes (27 bits) are required instead of the original six bytes (48 bits).

Table 3 Parameter compression bits in different scenarios

PC bits	Compression scenario
00	BF signal not compressed (sent completely)
01	BF signal fully compressed (not sent at all)
10	BF signal compressed (current value>previous value)
11	BF signal compressed (current value<previous value)

Table 4 Application of the BFC algorithm to the signals in Table 2

Parameter	Difference in value	PC (binary)	BPC (binary)	Signal type
Vehicle speed	14	10	–	BF
Coolant temp	−1	11	–	BF
Engine RPM	0	01	–	NBF
Fan RPM	3	–	0	NBF
Oil sensor	−2	–	0	NBF
Engine state	0	–	1	NBF
Gear state	0	–	1	NBF
Park state	0	–	1	NBF

Coolant temp: coolant temperature

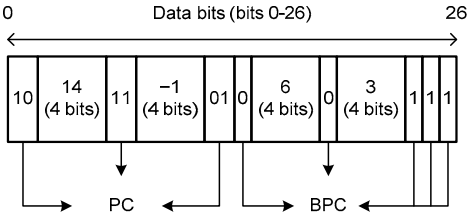

Fig. 4 Compressed data of the example in Table 2 obtained using the BFC algorithm

2.4 Compression area selection algorithm

In the compression area selection method, there is no need to predict the maximum value of the difference in successive CAN messages. In addition, the 64-bit data field is always assumed to be composed of eight signals, each with eight bits. This mechanism eliminates the need for the engineering standard of signal bit length.

After computing the difference values of signals between the current and preceding frames, each difference value is represented using nine bits. The magnitude of the difference is expressed using the most significant eight bits (bit 8 to bit 1) and the sign is denoted by the least significant bit (bit 0). If the difference is 0, the corresponding header bit is set to 0. Otherwise, the header bit is set to 1.

The header bits are placed at the last column beginning from the first row in a 2D map. Then starting from the next row, the non-zero difference values are placed from bit 8 to bit 1. Beginning from the leftmost column, a column is deleted if every element in the column is zero. The region from the column with the first non-zero element to the last column is selected as the data compression area.

Assume that we try to send eight signals {A, B, C, D, E, F, G, H}, each with eight bits. Also, assume that the difference values between the current frame and the previous frame are {0, 0, 2, 0, 0, −17, 0, 0}. Then '00100100' is sent as a header value with the actual difference values {2, −17}.

Table 5 shows the compression area selection map corresponding to this example. The selected bits (shaded) are rearranged according to the order given in Table 6. Compared with eight bytes of the original data before compression, the compressed data can be represented using only three bytes. Thus, in this example, the data compression rate is 62.5%.

Table 5 Compression area selection map

Signal		Bits 8–6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Header	H0							0
	H1							0
	H2							1
	H3							0
	H4							0
	H5							1
	H6							0
	H7							0
C diff. value		000	0	0	0	1	0	0
F diff. value		000	1	0	0	0	1	1

C/F diff. value: difference in value for signal C/F

Table 6 Memory map for the data in Table 5

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	H7[0]	H6[0]	H5[0]	H4[0]	H3[0]	H2[0]	H1[0]	H0[0]
Byte 1	F[3]	C[3]	F[2]	C[2]	F[1]	C[1]	F[0]	C[0]
Byte 2	0	0	0	0	F[5]	C[5]	F[4]	C[4]

3 The proposed CAN message compression scheme

In existing CAN data compression methods, the compression efficiency depends on the accuracy of the predicted maximum difference values (e.g., ±15 in the BFC algorithm). However, it is not easy to predict the maximum difference values of CAN signals accurately. In fact, compression efficiencies are improved if the maximum difference values can be adjusted depending upon system operating conditions.

In addition, if the data field of a CAN frame is composed of many different signals, the use of a DCC can impose a severe overhead. As an example, consider the EMS1 (engine management system 1) CAN signal in Table 7. In this case, if the BFC algorithm is used, 15 bits are required for the PC and BPC bits.

In the compression area selection algorithm, it is not necessary to predict the maximum value of the difference in successive CAN messages. However, the length of the data field is increased by up to eight bits due to the header bits.

In our proposed method, to overcome these problems, the 64-bit data field is assumed to be composed of three signals with 24, 24, and 16 bits, respectively. In addition, CAN signals are rearranged within the data field of a CAN frame based on the simulation of the actual CAN signals, such that the length of the reduced data field is minimized. Using the EMS1 CAN signals in Table 7, the signal rearrangement algorithm can be summarized as follows:

Table 7 EMS1 CAN signal (ID: 316)

Signal	Signal description	Bit length	Bit address
SWIGK	Key on status	1	0
FNENG	Engine speed signal error status	1	1
ACKTCS	Traction control system status	1	2
PUCSTAT	Fuel shut-off status	1	3
TQCORS	Torque adjustment status	2	4
RLYAC	Torque adjustment status	1	6
FSUBTQI	MEF (mass air flow) error status	1	7
CT	Current toque value	8	8
RPM	Engine speed value	16	16
IET	Engine torque indication	8	32
VS	Vehicle speed value	8	40

1. The rate of change of each signal in a CAN data field (e.g., Table 7) is estimated by computing the difference values between successive data fields using actual CAN data. Regardless of the original signal assignment, a data field is assumed to be composed of three signals. Since the number of data bits in Table 7 is 48, each of the three signals has a data length of 16 bits. Slowly changing signals are placed in the most significant parts and frequently changing CAN signals in the least significant parts of a 2D map (Table 8).

2. For each signal, compute the difference values of signals between the current and the preceding frames. Each difference value is then represented using a modified sign-magnitude number. That is, as opposed to the conventional sign-magnitude number, the sign is denoted by the least significant bit. The length of each modified sign-magnitude number is one larger than the length of each signal.

Table 8 EMS1 CAN signal rearrangement

Signal*	Bit address	
	Bits 15–8	Bits 7–0
1	CT[7:0]	SWIGK[0]–FSUBTQI[0]
2	IET[7:0]	RPM[7:0]
3	RPM[7:0]	VS[7:0]

* 16 bits

3. Since the number of the CAN signals in a data field is three, three header bits are used. If the difference values of a signal are all zeroes, the corresponding header bit is set to 0. Otherwise, it is set to 1.

4. The header bits are placed at the last column beginning from the first row in a 2D map (compression area selection map). Then starting from the next row, the non-zero difference values are placed.

5. Beginning from the leftmost column, a column is deleted if every element in the column is zero. The region from the column with the first non-zero element to the last column is selected as the compression area.

As an example, assume that we try to send EMS1 CAN signals. First, Table 8 is obtained by calculating the rate of change of actual CAN signals. Then based on the simulation result, the CAN signals to be transmitted are rearranged according to Table 8.

Assume that the three signal values in the previous frame are {31 745, 11 832, 15 360} and that these values in the current frame are {31 740, 11 849, 15 360}. The difference values are {−5, 17, 0}. Thus, the header bits are '110' (Table 9).

Table 9 Example of difference values

Signal	Number of previous frames	Number of current frames	Difference	Header bit
1	31 745	31 740	−5	1
2	11 832	11 849	17	1
3	15 360	15 360	0	0

Table 10 shows the compression area selection map. The selected bits (shaded) are rearranged according to the order given in Table 11. If all three header bits are zero, there is no need to send any data. In this case, the DLC is set to zero. Compared with six bytes of the original data before compression, the compressed data can be represented using only two bytes (Table 11). Thus, in this example, we can achieve a data compression rate of 66.6%.

In the proposed algorithm, if the received DLC value is the same as the predetermined DLC value, the received frame contains the original (non-compressed) data. Otherwise, for a decreased DLC value, the receiving unit is notified that the received CAN frame contains compressed data. Thus, in the proposed algorithm, the use of two message IDs is avoided (Miucic *et al.*, 2009).

Table 10 Compression area selection map

Signal		Bits 8–6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Header	H0							1
	H1							1
	H2							0
Signal 1 (S1)		000	0	0	1	0	1	1
Signal 2 (S2)		000	1	0	0	0	1	0

Table 11 Memory map for the data in Table 10

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	S1[2]	S2[1]	S1[1]	S2[0]	S1[0]	H2[0]	H1[0]	H0[0]
Byte 1	0	S2[5]	S1[5]	S2[4]	S1[4]	S2[3]	S1[3]	S1[2]

Note that the size of the memory map is determined by the size of the data compression area. Thus, it is not necessary to determine the size of the memory map in advance. In other words, it is not necessary to predict the maximum difference values. Consequently, we can avoid the inefficient data compression caused by inaccurate prediction of the maximum difference values. Figs. 5 and 6 show flowcharts of the proposed

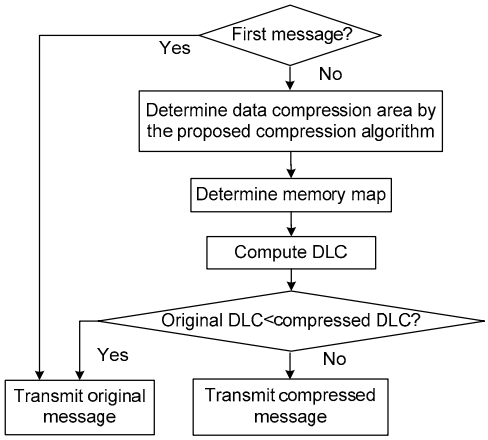

Fig. 5 Flow chart of the proposed compression algorithm

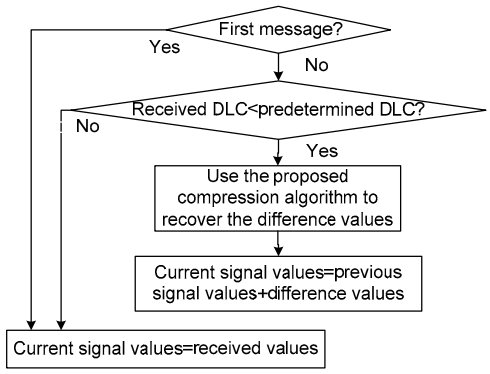

Fig. 6 Flow chart of the proposed recovery algorithm

CAN data compression and recovery methods, respectively.

4 Performance analysis

In this section, the performance of the proposed method is compared with those of the EDR, BFC, and compression area selection algorithms. The CAN signals used for the simulation were EMS1 (Table 7), EMS2 (Table 12), EMS4 (Table 13), and TCU1 (transmission control unit 1) (Table 14) signals.

The simulation signals were obtained from actual driving of a test vehicle. Fig. 7 shows the original data and recovery data of EMS1 CAN RPM signal.

Table 12 EMS2 CAN signal (ID: 329)

Signal	Signal description	Bit length	Bit address
MULINFO	Multiplexed information	6	0
MULCODE	Identification of information	2	6
TEMPENG	Engine coolant temperature	8	8
MAFFAC	Mass air flow correction factor	8	16
VBOFF	ECU adaptive values	1	24
ACKES	Acknowledgement, engine stopped	1	25
CONF	Configuration of MIL HFMM	3	26
Free	Free	1	29
ACCACT	Auto cruise control in activation	1	30
EXHGAS	Request for exhaust gas pattern	1	31
BRAKE	Indication of brake switch ON/OFF	2	32
ENGCHR	Engine characteristic (kind of fuel)	4	34
Free	Free	2	38
TPS	Throttle angle	8	40

Table 13 EMS4 CAN signal (ID: 545)

Signal	Signal description	Bit length	Bit address
Free	Free	1	0
LMIL	Lamp ‘check engine for OBD’	1	1
IMSTAT	Status ‘immobilizer’	1	2
AMPCAN	Atmospheric pressure	5	3
FCO	Fuel consumption	16	8
VB	Battery voltage	8	24
TQIACORJ	Actual engine torque	16	32

Comparisons of the compression efficiencies of the CAN signals obtained under normal and sudden stop/acceleration driving conditions are illustrated in Tables 15 and 16, respectively. It is observed that we can obtain an additional compression efficiency of 0–22% with the proposed method, compared with other CAN data compression methods.

Fig. 8 shows the comparison of ID 43F CAN data compression efficiencies under normal driving. In most cases, six bytes of CAN data are compressed to one byte using the proposed method. However, with other algorithms, in most cases six bytes of CAN data are compressed to one or two bytes.

Table 14 TCU1 CAN signal (ID: 43F)

Signal	Signal description	Bit length	Bit address
TARGC	Target of gear change	3	0
SWIGS	Gear change active	1	3
FOBD	OBD-relevant error in TCU	1	4
TCUSTAT	Status TCU	1	5
SWICC	Converter lockup clutch	2	6
GSELDISP	Gear selector display	4	8
FTCU	TCU fault	2	12
TCUTYPE	Control unit type	2	14
TCUOBD	OBD status, transmission control	4	16
ISASQ21	Downshift from the 2nd to 1st gear active	1	20
Free	Free	3	21
TQITCU	Torque intervention of TCU	8	24
TEMPAT	A/T fluid temperature	8	32
NTC	Torque converter turbine speed	8	40

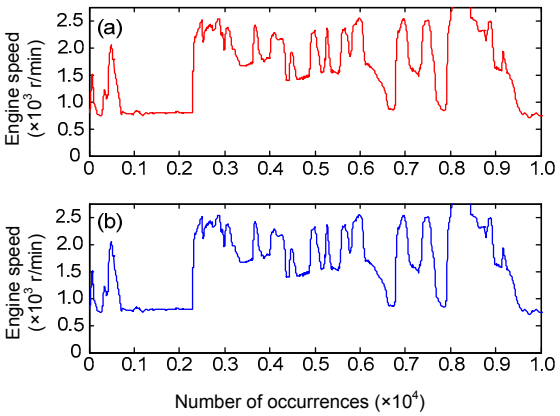

Fig. 7 Actual CAN signals
(a) Original data; (b) Recovery data

Table 15 Comparison of the compression efficiencies under normal driving conditions

Compression method	Number of transmitted data			
	ID: 316	ID: 329	ID: 43F	ID: 545
EDR	1 370 496 (54.62%)	1 069 776 (64.58%)	1 233 504 (66.62%)	1 527 824 (49.41%)
BFC	1 321 760 (56.24%)	1 012 376 (66.48%)	1 269 048 (65.66%)	1 789 864 (40.74%)
Compression area selection	900 376 (70.19%)	582 240 (80.72%)	865 280 (76.58%)	1 184 336 (60.79%)
Proposed	705 016 (76.66%)	571 944 (81.06%)	654 504 (82.29%)	1 138 864 (62.29%)

Total numbers of transmitted data are 3 020 160 (ID: 316), 3 020 112 (ID: 329), 3 695 280 (ID: 43F), and 3 020 112 (ID: 545), respectively. The values in the brackets represent the compression efficiencies with respect to the corresponding total number of transmitted data

Table 16 Comparison of the compression efficiencies under sudden stop/acceleration

Compression method	Number of transmitted data			
	ID: 316	ID: 329	ID: 43F	ID: 545
EDR	96 464 (50.60%)	72 016 (63.09%)	66 792 (65.78%)	76 688 (60.71%)
BFC	92 040 (52.86%)	65 808 (66.27%)	79 536 (59.25%)	79 152 (59.44%)
Compression area selection	67 688 (65.34%)	40 936 (79.02%)	59 648 (69.44%)	51 296 (73.72%)
Proposed	53 176 (72.77%)	39 992 (79.50%)	46 216 (76.32%)	45 120 (76.88%)

Total numbers of transmitted data are 195 264 (ID: 316), 195 120 (ID: 329), 195 168 (ID: 43F), and 195 168 (ID: 545), respectively. The values in the brackets represent the compression efficiencies with respect to the corresponding total number of transmitted data

Fig. 9 shows the hardware implementation of the CAN compression algorithm using a Cortex M3 embedded board. Fig. 10 shows the test environments,

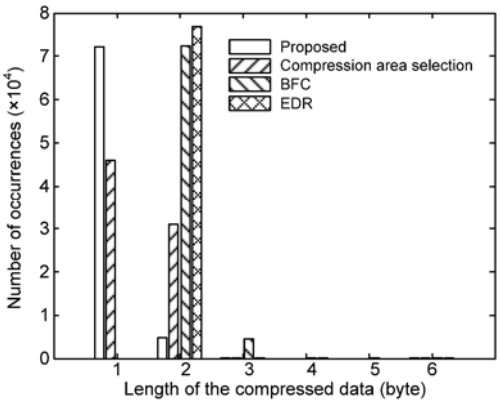

Fig. 8 Comparison of ID 43F CAN data compression efficiencies under normal driving

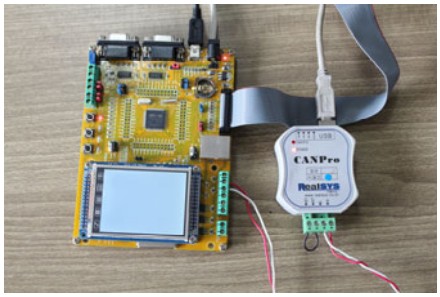

Fig. 9 Cortex M3 embedded test board

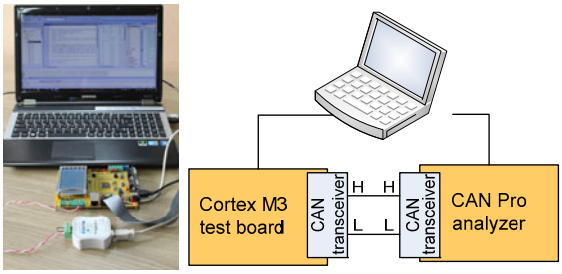

Fig. 10 Test environments

including the CAN Pro analyzer. By using a Cortex M3 embedded test board, 64-bit EMS CAN data compression can be performed within 0.16 ms; consequently, the proposed algorithm is suitable for automobile applications since EMS CAN signals are usually transmitted every 10 ms in automobiles.

5 Conclusions

In this paper, a CAN message compression method is presented. As opposed to EDR, BFC, and compression area selection algorithms, with the proposed method, it is not necessary to predict the maximum difference values in successive CAN messages. In addition, the number of header bits is limited to up to three, regardless of the number of

signals in the data field of a CAN frame. Through simulations using actual CAN data, we have shown that the CAN transmission data are further reduced by up to 22% with the proposed method, compared with conventional methods.

With an embedded test board, CAN data compression can be performed within 0.16 ms and, consequently, the proposed algorithm is suitable for use in automobile applications.

References

Bosch, 1991. CAN Specification Version 2.0. Stuttgart, Germany.

Desai, M., Shetty, R., Padte, V., *et al*., 2013. Controller area network for intelligent vehicular systems. Proc. Int. Conf. on Advances in Technology and Engineering, p.1-6. [doi:10.1109/ICAdTE.2013.6524757]

ISO (International Organization for Standardization), 2003. Road Vehicles—Controller Area Network (CAN)—Part 1: Data Link Layer and Physical Signalling, ISO 11898-1: 2003.

Kelkar, S., Kamal, R., 2014. Boundary of fifteen compression algorithm for controller area network based automotive applications. Proc. Int. Conf. on Circuits, Systems, Communication and Information Technology Applications, p.162-167. [doi:10.1109/CSCITA.2014.6839253]

Lawrenz, W., 1997. CAN System Engineering: from Theory to Practical Applications. Springer, New York.

Leen, G., Heffernan, D., 2002. Expanding automotive electronic systems. *Computer*, **35**(1):88-93. [doi:10.1109/2.976923]

Misbahuddin, S., Mahmud, S.M., Al-Holou, N., 2001. Development and performance analysis of a data-reduction algorithm for automotive multiplexing. *IEEE Trans. Veh. Technol.*, **50**(1):162-169. [doi:10.1109/25.917911]

Miucic, R., Mahmud, S., 2006. An improved adaptive data reduction protocol for in-vehicle networks. SAE Tech. Paper, 2006-01-1327. [doi:10.4271/2006-01-1327]

Miucic, R., Mahmud, S.M., Popovic, Z., 2009. An enhanced data-reduction algorithm for event-triggered networks. *IEEE Trans. Veh. Technol.*, **58**(6):2663-2678. [doi:10.1109/TVT.2008.2011361]

Ortega, E., Heurung, T., Swanson, R., 2006. System design from wires to warranty. *Automot. Electron. Mag.*, p.14-18.

Ramteke, P., Mahmud, S., 2005. An adaptive data-reduction protocol for the future in-vehicle networks. SAE Tech. Paper, 2005-01-1540. [doi:10.4271/2005-01-1540]

Wu, Y.J., Chung, J.G., Sunwoo, M.H., 2014. Design and implementation of CAN data compression algorithm. Proc. IEEE Int. Symp. on Circuits and Systems, p.582-585. [doi:10.1109/ISCAS.2014.6865202]