2013년 11월 전자공학회 논문지 제 50 권 제 11 호
Journal of The Institute of Electronics Engineers of Korea Vol. 50, NO. 11, November 2013 http://dx.doi.org/10.5573/ieek.2013.50.11.099

논문 2013-50-11-13

DLC와 전송 데이터 압축영역 설정을 이용한 CAN 데이터 압축

(CAN Data Compression Using DLC and Compression Area Selection)

오 유 경*, 정 진 균**

(Yujing Wu and Jin-Gyun Chung)

요 약

Controller Area Network (CAN)의 개발 목적은 차량 내 Electronic Control Units (ECUs)간의 다중 통신을 통해 자동차에서 큰 부피와 무게를 차지하는 와이어 하네스를 저비용의 네트워크 케이블로 대체하기 위한 것이었다. 차량에 탑재되는 ECU들의 증가로 인해 CAN 데이터 전송량이 많아짐에 따라 CAN 버스로드와 오류 확률도 증가하고 있다. CAN 데이터 전송 시간은 CAN 프레임의 길이에 비례하기 때문에 프레임의 길이를 줄이게 되면 효율적으로 CAN 버스로드와 오류확률을 감소시킬 수 있다. 본 논문에서는 CAN 메시지 길이를 감소시키기 위해 Data Length Code(DLC)와 전송 데이터 압축영역 설정 절차를 사용한 CAN 메시지 압축 알고리즘을 제안한다. 제안한 방법에서는 기존의 알고리즘과 달리 변화량을 저장하기 위한 최대 변화량의 범위를 설정하지 않아도 되기 때문에 부정확한 설정에서 발생하는 오류나 지나친 설정에서 발생하는 압축효율 저하를 피할 수 있다. 또한, DLC 크기에 의해 압축 유무를 판단함으로써 기존 방법에서 제안된 두 개의 ID로 압축 여부를 판단하는 비효율적인 문제점을 해결할 수 있다. 실제차량 주행 후 얻은 데이터로 시뮬레이션 해본 결과, 기존의 방법에 비해 최대 52%까지 더 압축된 것을 확인하였다. 또한, 임베디드 테스트 보드를 이용하여 테스트 했을 때 한 개의 64비트 EMS CAN 데이터를 압축하는데 0.16ms가 소요되어 차량용 CAN 통신에 사용가능함을 보인다.

Abstract

Controller area network (CAN) was designed for multiplexing communication between electronic control units (ECUs) in vehicles and thus for decreasing the overall wire harness. The increasing number of ECUs causes the CAN bus overloaded and consequently the error probability of data transmission increases. Since the time duration for the data transmission is proportional to CAN frame length, it is desirable to reduce the frame length. In this paper, a CAN message compression method is proposed using Data Length Code (DLC) and compression area selection algorithm to reduce the CAN frame length and the error probability during the transmission of CAN messages. By the proposed method, it is not needed to predict the maximum value of the difference in successive CAN messages as opposed to other compression methods. Also, by the use of DLC, we can determine whether the received CAN message has been compressed or not without using two ID's as in conventional methods. By simulations using actual CAN data, it is shown that the CAN transmission data is reduced up to 52 % by the proposed method, compared with conventional methods. By using an embedded test board, it is shown that 64bit EMS CAN data compression can be performed within 0.16ms and consequently the proposed algorithm can be used in automobile applications without any problem.

Keywords : CAN, DLC, Embedded System, Data Compression

* 학생회원, ** 정회원, 전북대학교 전자공학부
(Department of Electronic Engineering, Chonbuk National University)
※ This work was supported by the IT R&D program of MOTIE/KEIT. [10044092, Development of Core IPs of OFDM PHY and RF Transceiver for 60GHz Wireless LAN/PAN in application of 7Gbps Wireless Multimedia Services]
ⓒ Corresponding Author(E-mail: jgchung@jbnu.ac.kr)
접수일자 : 2013년8월16일, 수정완료일 : 2013년10월24일

Ⅰ. 서 론

자동차 부품 업체인 Bosch사에서 1986년에 자동차 내부의 서로 다른 3개의 전자장치(ECU, electronic control unit)간의 통신을 위한 통신 장치로서 차량 네트워크용 CAN을 최초로 개발하였다. CAN 프로토콜은 ABS와 에어백 등을 제어하기 위하여 빠른 응답성과 높은 신뢰성을 갖고 있으며 고수준의 보안 기능을 갖춘 실시간 직렬 전송 프로토콜이다[1~2]. 최초의 CAN 프로토콜 컨트롤러 칩이 제작된 후 20여년이 지난 지금도 CAN은 그 사용이 계속 증가되고 있다.

또한, 차량용 외에도 CAN의 안정성과 가격경쟁력이 입증되면서 로봇, 공장자동화, 비행기, 의료기기 등 다양한 산업분야에 적용되고 있다. 많은 마이크로컨트롤러 제조업체들은 CAN 인터페이스를 통합한 솔루션을 제공하며 통합된 CAN 컨트롤러의 성능과 구조의 최적화가 계속되고 있다.

자동차 한 대당 적게는 25개, 고급차종인 경우에는 100여개의 ECU가 탑재된다[3]. 자동차의 ECM (Engine Control Module)을 비롯하여 트랜스미션, 에어백, ABS 등에 사용되는 70여개의 ECU 데이터는 전체 차량 데이터의 약 40%를 차지한다. 차량에 연결된 ECU 개수의 증가는 CAN 버스 점유율을 증가시킬 뿐만 아니라 메시지 전송오류 확률의 증가를 초래한다. 또한, 버스로드의 증가로 인해 우선순위가 낮은 메시지는 전송지연시간이 길어지게 된다[4~5].

CAN 메시지의 길이를 단축시키면 CAN 버스로드를 효과적으로 감소시킬 수 있다. CAN 메시지의 변화가 매 프레임마다 크지 않다는 점에 착안하여 이전 프레임과 현재 프레임의 변화량만 전송하면 CAN 메시지 데이터의 길이는 현저하게 감소시킬 수 있다. 하지만 차이 값만 전송하려면 최대변화량의 범위를 정하여야 하는데, 최대변화량의 범위를 정하게 되면 작은 차이 값에도 할당된 비트를 모두 사용해야 하는 비효율성이 발생하게 된다[6]. 비트재배열 알고리즘을 적용하면 최대변화량을 정하지 않아도 되지만 전송 데이터의 변화량이 0인 경우가 자주 발생하면 비효율성이 증가하게 된다[7].

본 논문에서는 CAN 메시지 길이를 감소시키기 위해 DLC와 전송 데이터 압축영역 설정 절차를 사용한 CAN 메시지 압축 알고리즘을 제안한다. CAN 압축 알고리즘을 Cortex M3 임베디드 시스템으로 구현하였으며 정상주행 시 평균 72%이상의 압축률을 보이고 과속/급정거가 자주 발생하는 운행 시 기존의 방법에 비해 52% 정도 더 압축효율이 증가함을 보인다.

Ⅱ. CAN 데이터 프레임

표준 CAN 2.0A에서는 11비트 식별자를 사용하고, 2.0B에서는 29비트 식별자를 사용한다. CAN 노드에서 메시지를 보내기 전에 CAN 버스 라인이 사용 중인지를 파악하고 사용 중이지 않으면 전송하려고 했던 CAN 노드들은 각자의 메시지 식별자를 보내기 시작한다. CAN 버스 중재 과정은 우선순위가 낮은 메시지가 자동적으로 메시지 전송을 중단하고 다음 버스 사이클에서 재전송을 수행한다. 하지만 이때 까지도 높은 우선순위의 메시지 전송이 완료되지 않으면, 전송을 완료할 때까지 대기해야 한다.

11비트 식별자를 갖는 표준 CAN Data Frame은 그림 1과 같이 구성된다. 표준 CAN의 Data Frame은 총 7개의 비트 구간으로 나눌 수 있다. 데이터 필드는 한 노드로부터 다른 노드로 전하고자 하는 데이터를 포함하여 0~8byte 로 구성된다.

데이터 필드는 최대 64bit로서 ID에 따라 길이가 다르며 CAN 프레임 내에서 가장 많은 비트 수를 포함하고 있고 가장 효율적으로 압축될 수 있는 영역이다. 최대 64bit 로 구성된 데이터 필드를 2차원 메모리로 간주할 때 메모리를 byte 0~7과 각각의 비트들로 구분하여 표 1과 같이 총 64비트로 표현할 수 있다.

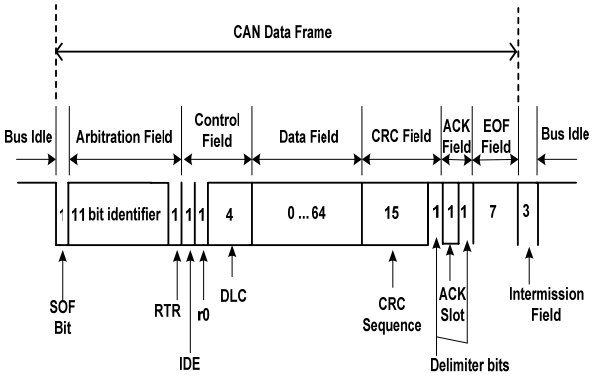

그림 1. 표준 CAN 메시지 구조.
Fig. 1. Format of the CAN 2.0A data frame.

표 1. 64-비트 CAN 메시지 메모리 맵
Table 1. Memory map of 64-bit CAN message.

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
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

Ⅲ. 기존의 CAN 데이터 압축 알고리즘

1. 능동형 CAN 데이터 압축 알고리즘

차량 내 한 CAN 노드에서 데이터 전송은 보통 10ms 주기로 이루어지는데, 이는 운전자가 차량을 조작하는 시간에 비하면 훨씬 작은 시간 주기이다. 따라서 운전자의 차량운전 시간을 고려하면 이전 데이터 프레임과 현재 데이터 프레임의 변화량은 별로 크지 않다.

CAN 메시지의 변화가 매 프레임마다 크지 않다는 점에 착안하여 이전 프레임과 현재 프레임의 변화량만 전송하면 CAN 메시지 데이터의 길이는 현저하게 감소시킬 수 있으며 대표적인 방법으로 능동형 압축방법과 비트재배열을 이용한 방법이 있다.

능동형 압축 방법에서는 압축메시지의 데이터 정렬 순서 및 크기를 미리 정의하고 두 개의 ID로 데이터 압축 여부를 판별한다. 변화량 크기가 설정된 변화량 범위를 초과하게 되면 원래메시지 ID로 CAN 메시지를 전송하고 압축메시지 ID는 해당 ID에서 1을 뺀 값으로 한다[6]. 예를 들어 압축하지 않은 메시지의 ID가 100일 경우 압축한 메시지의 ID는 99로 한다.

64비트의 Data Field가 각각 8비트를 가지는 8개의 신호 {A, B, C, D, E, F, G, H}로 구성되고 ID가 100인 경우를 예를 들어 설명한다. 만약 이전 프레임의 {A, B, C, D, E, F, G, H}가 {5, 15, 58, 0, 15, 60, 0, 34}이었고 현재 프레임에는 {5, 10, 60, 0, 15, 40, 0, 40}으로 변하였다면 차이 값은 표 2와 같이 표현 가능하다.

{A, B, C, D, E, F, G, H}의 신호 차이 값이 {0, -5, 2, 0, 0, -20, 0, 6}이면 헤더는 01100101을 전송하고 실제 변화된 값 {-5, 2, -20, 6}만 전송한다. 데이터가 전송되면 수신부에서는 헤더의 01100101을 분석하여 이전 값 {A, D, E, G}는 그대로 두고 {B, C, F, H}에 수신된 값 {-5, 2, -20, 6}를 더하여 각각의 신호 값을 복원한다. 표 3과 같이 전송해야할 헤더비트들과 변화량을 2차원 메모리 맵에 맵핑하게 되면 원래의 64비트에서 40비트로 37.5%가 압축됨을 알 수 있다. 능동형 압축 방법에서는 한 ID의 CAN 데이터 압축을 위해 두 개의 ID를 사용해야하므로 더 많은 메모리와 처리시간이 필요하게 된다.

2. 비트 재배열을 이용한 CAN 데이터 압축 알고리즘

능동형 CAN 데이터 압축 방법에서는 차이 값이 0인 경우 데이터가 전송되지 않으므로 버스로드가 감소하지만 최대 변화량 예측이 난해한 경우 최대 변화량을 확장하여야 하므로 압축효율이 감소하게 되는 단점이 있다. 또한, 작은 차이 값 전송에도 이미 할당된 변화량 비트들을 모두 사용해야하는 비효율성이 발생하게 된다.

표 2. 능동형 방법에서 전송할 데이터의 변화량 및 헤더 비트
Table 2. Transmission data of difference and header bit by Active method.

신호	A	B	C	D
이전 신호 값	5	15	58	0
현재 신호 값	5	10	60	0
차이 값	0	-5	2	0
변화량(binary)	0	11111011	00000010	0
헤더 비트	0	1	1	0
신호	E	F	G	H
이전 신호 값	15	60	0	34
현재 신호 값	15	40	0	40
차이 값	0	-20	0	6
변화량(binary)	0	11101100	0	00000110
헤더 비트	0	1	0	1

표 3. 능동형 방법의 CAN 데이터 압축 메모리 맵
Table 3. CAN data compression memory map by Active method.

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
byte 0	헤더							
byte 1	B신호의 변화량							
byte 2	C신호의 변화량							
byte 3	F신호의 변화량							
byte 4	H신호의 변화량							

비트 재배열을 이용한 CAN 데이터 압축 방법에서는 각 데이터 차이 값을 원 신호와 같은 비트수를 할당하여 표시하고 최하위 비트는 부호 비트로 정의한다[7]. 차이 값이 음수일 때 변화량의 최하위 비트는 1로, 양수일 때 변화량의 최하위 비트를 0으로 설정한다. 예를 들면 표 2와 같이 B신호의 차이 값이 -5일 때 변화량은 000001011로 표시하고 C신호와 같이 차이 값이 2일 때 변화량은 000000100으로 표시한다. 각 신호들의 변화량을 표 4와 같이 테이블 형태로 정열한 후 MSB 열부터 시작하여 한 열의 해당 비트들이 모두 0인 경우해당 열을 제거한다. 처음으로 제거되지 않고 남은 bit 4 열부터 시작하여 bit 0열까지를 압축 데이터 구성영역으로 결정하고 이 데이터들만을 전송한다.

표 4의 맨 마지막 열의 모든 bit들 부터 시작하여 압축 데이터 구성영역에 있는 모든 데이터들을 표 5의 2차원 메모리 맵에 첫 번째 행부터 순서대로 채워 넣으면 원래의 64비트에서 40비트로 37.5%의 압축률을 보이게 된다.

비트 재배열을 이용한 CAN 데이터 압축 방법에서는 최대 변화량의 크기를 설정하지 않기 때문에 작은 차이 값 전송에도 이미 할당된 변화량 비트들을 모두 사용해야 하는 비효율성 문제를 해결할 수 있다. 하지만 변화량의 상위 비트에서 1이 나타나면 압축 데이터 구성영역이 확장되어 필요 이상으로 많은 신호들의 변화량 비트들을 전송해야 하는 비효율성이 발생하게 된다.

표 4. 비트 재배열을 이용한 CAN 데이터 압축 데이터 구성영역
Table 4. Compression area by Variable length CAN data compression algorithm.

신호	bit7~5	bit4	bit3	bit2	bit1	bit0
A	0...0	0	0	0	0	0
B	0...0	0	1	0	1	1
C	0...0	0	0	1	0	0
D	0...0	0	0	0	0	0
E	0...0	0	0	0	0	0
F	0...0	1	0	1	0	1
G	0...0	0	0	0	0	0
H	0...0	0	1	1	0	0

표 5. 표4의 전송 데이터 맵핑 순서
Table 5. Mapping procedure for the data in Table 4.

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
byte 0	0	0	1	0	0	0	1	0
byte 1	0	0	0	0	0	0	1	0
byte 2	1	0	1	0	0	1	0	0
byte 3	1	0	0	0	0	0	1	0
byte 4	0	0	1	0	0	0	0	0

Ⅳ. 제안한 CAN 데이터 압축 알고리즘

1. 제안한 CAN 데이터 압축 알고리즘

능동형 CAN 압축 알고리즘은 최대 변화량 예측이 난해한 경우 최대 변화량을 확장하여야 하므로 압축효율이 감소하게 되는 단점이 있고 압축유무를 판별함에 있어 두 개의 ID를 사용하여 더 많은 메모리와 처리시간을 요구하게 된다. 비트재배열 방법에서는 한 신호의 변화량의 크기가 클 때에는 변화량이 0인 신호를 포함하여 모든 신호들의 변화량 비트들을 모두 전송해야 하는 비효율성 문제가 존재하게 된다. 이러한 문제를 개선하기 위해 DLC(Data Length Code)에 의해 압축 유무를 판단하고 전송데이터 압축영역 설정 절차 및 헤더를 이용한 압축방법을 제안한다.

현재 자동차 회사마다의 기술 표준이 다름에 따라 ID에 따른 신호들과 그 신호들의 비트 address 및 비트 사이즈들이 서로 다르기 때문에 기존의 방법들로 압축을 진행하려면 자동차 기술 표준이 요구된다. 기술 표준에 의하여 압축을 진행한다고 하여도 ID에 따른 신호들의 개수가 많고 신호들의 비트 사이즈들이 불규칙적이면 더 많은 처리시간이 필요하게 된다. 또한 헤더를 사용한다면 신호들의 개수 증가에 따라 헤더의 비트수 증가를 초래하여 헤더에 의한 오버헤드가 증가하게 된다.

본 논문에서는 64비트의 데이터 필드를 각각 8비트를 가지는 8개의 신호로 간주하여 기술 표준에 의한 신호들의 address와 사이즈를 구별할 필요성을 제거한다. 압축된 데이터의 DLC를 계산하고 압축된 데이터의 DLC가 원래 데이터의 DLC보다 작은 경우 압축된 데이터를 전송하고 그렇지 않은 경우에는 원래의 압축되지 않은 데이터를 전송한다.

수신 측에서는 미리 정의 되어 있는 각 메시지 ID의 DLC를 체크하여 전송된 CAN 프레임의 압축여부를 판단할 수 있게 된다. 따라서 두 개의 ID로 압축 여부를 판단하는 능동형 압축 방법의 비효율적인 문제점을 해결할 수 있다. 압축을 위해서는 전송 데이터 압축영역 설정절차를 통해 전송할 데이터들의 압축영역을 설

정한다.

전송 데이터 압축영역 설정 절차

1. 한 신호의 변화량을 9비트로 표현하고 상위 8비트는 차이 값의 크기, 최하위 비트는 부호 비트로 정의한다. 차이 값이 음수일 때 변화량의 최하위 비트는 1로, 양수일 때 변화량의 최하위 비트를 0으로 설정한다.
2. 변화량이 존재하는 신호들의 헤더 비트를 1로, 변화량이 0인 신호들의 헤더 비트를 0으로 설정한다.
3. 헤드 비트들을 LSB 열에 하나의 열로 먼저 배열하고 헤더비트가 1에 해당하는 9개의 비트들을 모든 헤더비트의 나열이 끝난 다음 하나의 행으로 배열한다.
4. 위 단계들을 통해 배열된 표에 대해 MSB 열부터 시작하여 한 열의 해당 비트들이 모두 0인 경우 해당 열을 제거한다.
5. 처음으로 제거되지 않고 남은 열부터 시작하여 LSB까지의 열들을 압축 데이터 구성영역으로 결정한다.

예를 들면 표 6과 같이 B신호의 차이 값이 -5일 때 변화량은 000001011로 설정하고 C신호와 같이 차이 값이 2일 때 변화량은 000000100으로 설정한다. 표 6에서 변화량이 0이 아닌 신호들의 헤더 비트를 1로 설정하면 헤더의 2진수 표현은 01100101이다. 헤더 비트가 1인 4개 신호의 비트들을 표 7과 같이 배열한다. 표 8은 표 7의 전송 데이터 비트들을 재배열하고 메모리 맵에 맵핑하는 순서를 보여준다. 표 9는 표 8의 맵핑순서에 의해 메모리 맵에 데이터를 맵핑한 결과이다.

기존의 방법에서 메모리 맵의 크기는 미리 정의한 최대변화량 범위에 의해 결정된다. 하지만 제안한 방법에서 데이터 메모리 맵의 크기는 데이터 압축 영역에 의해 결정된다. 즉, 제안한 방법에서는 변화량의 최대값과 무관하게 압축을 수행하므로 기존의 방법에서 발생하는 최대변화량의 불완전한 예측으로 인해 발생하는 오류나 압축효율감소 현상을 피할 수 있다.

그림 2는 제안한 CAN 메시지 압축 순서도이며 그림 3은 복원 순서도이다. 그림 4는 제안한 알고리즘에 의해 EMS CAN의 RPM 신호를 압축 및 복원하였을 때 정확히 일치함을 보인다.

표 6. 제안한 방법에 의한 전송할 데이터의 변화량 및 헤더 비트
Table 6. Transmission data of difference and header bit by proposed method.

신호	A	B	C	D
차이 값	0	-5	2	0
변화량(binary)	0	000001011	000000100	0
헤더 비트	0	1	1	0
신호	E	F	G	H
차이 값	0	-20	0	6
변화량(binary)	0	000101001	0	000001100
헤더 비트	0	1	0	1

표 7. 표 6의 전송 데이터에 대한 압축영역
Table 7. Compression area selection for the data in Table 6.

신호	bit8~5	bit4	bit3	bit2	bit1	bit0
헤더						0
						1
						1
						0
						0
						1
						0
						1
B	0...0	0	1	0	1	1
C	0...0	0	0	1	0	0
F	0...0	1	0	1	0	1
H	0...0	0	1	1	0	0

표 8. 표 7의 전송 데이터 맵핑 순서
Table 8. Mapping procedure for the data in Table 7.

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
byte 0	헤더 [7]	헤더 [6]	헤더 [5]	헤더 [4]	헤더 [3]	헤더 [2]	헤더 [1]	헤더 [0]
byte 1	H[1]	F[1]	C[1]	B[1]	H[0]	F[0]	C[0]	B[0]
byte 2	H[3]	F[3]	C[3]	B[3]	H[2]	F[2]	C[2]	B[2]
byte 3	0	0	0	0	H[4]	F[4]	C[4]	B[4]

표 9. 압축 후 전송 데이터 메모리 맵
Table 9. The memory map of compressed data.

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
byte 0	1	0	1	0	0	1	1	0
byte 1	0	0	0	1	0	1	0	1
byte 2	1	0	0	1	1	1	1	0
byte 3	0	0	0	0	0	1	0	0

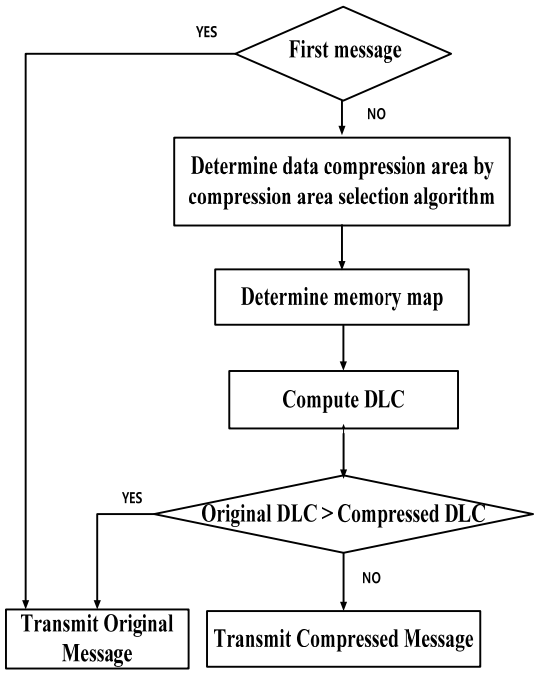

그림 2. 제안한 방법의 CAN 메시지 압축 순서도.
Fig. 2. The flowchart of the proposed compression method.

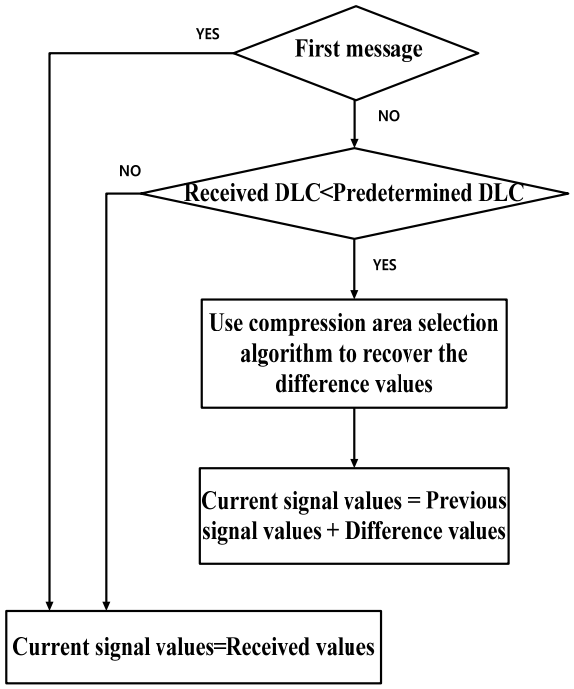

그림 3. 제안한 방법 CAN 메시지 복원 순서도.
Fig. 3. The flowchart of compressed data recovery.

표 10과 표 11은 각각 EMS(Engine Management System) CAN 신호 및 TCU2(Transmission Control Unit) CAN 신호의 종류와 비트길이를 나타낸다. 표 12와 표 13은 능동형과 재배열 및 제안한 방법을 이용하여 실제 차량 CAN 로깅파일로 MATLAB 시뮬레이션에 의해 얻어진 결과이다. 표 12에서 보인바와 같이 EMS CAN 신호의 경우 정상주행 시 제안한 방법은 능동형 방법 및 비트재배열 방법에 비해 각각 12.91%와 19.37%의 추가적인 압축효율을 얻을 수 있다. 과속/급정거 시에는 10.79%와 18.35%의 추가적인 압축효율을 얻을 수 있다. 표 13에서 보인 바와 같이 TCU2 CAN 신호의 경우 과속/급정거 시에는 제안한 방법으로 기존의 압축 알고리즘에 비해 전송 데이터양이 52% 더 감소한 것을 알 수 있다.

그림 5는 정상주행 시 제안한 방법과 능동형 방법에 의한 EMS CAN 데이터의 압축 전송 효율 비교도표이다. 그림 5와 같이 능동형 방법으로 압축한 결과 8byte CAN 데이터는 주로 2, 3 또는 4byte로 압축된 반면 제안한 방법으로 압축하면 주로 1또는 2byte로 압축된다.

그림 6은 정상주행 시 제안한 방법과 능동형 방법에 의한 TCU2 CAN 데이터의 압축 전송 효율 비교도표이다. 그림 7은 과속/급정거 시 제안한 방법과 능동형 방법에 의한 TCU2 CAN 데이터의 압축 전송 효율 비교도표이다.

그림 7에서 볼 수 있듯이 데이터의 변화량이 많은 과속/급정거 상황에서는 제안한 방법으로 압축하게 되면 기존의 방법에 비해 약 52%의 추가적인 압축효률을 얻을 수 있다.

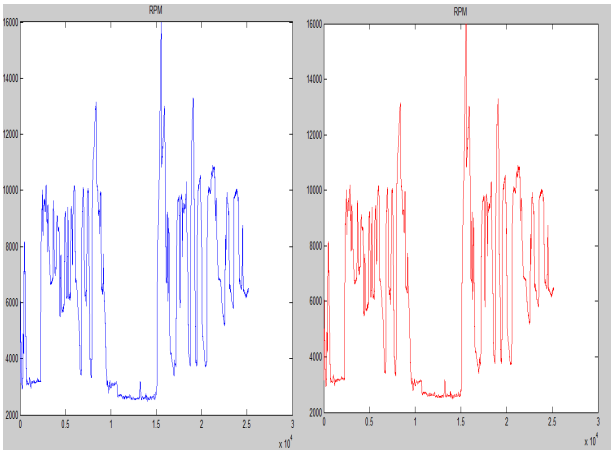

그림 4. EMS CAN의 RPM 신호 압축 및 복원.
Fig. 4. EMS CAN RPM signal compression and recovery.

표 10. EMS CAN 신호
Table 10. EMS CAN signal.

신호	신호 설명	비트길이
SWI_IGK	key on 상태	1
F_N_ENG	engine speed signal 이 error 상태	1
ACK_TCS	TCS 상태	1
PUC_STAT	연료 차단 상태	1
TQ_COR_STAT	토크 조정 상태	2
RLY_AC	에어컨 압력기 동작 상태	1
F_SUB_TQI	MEF 이상 상태	1
CT	현재 토크 값	8
RPM	RPM 속도 값	16
IET	엔진 토크 지시 값	8
FT	토크 저항 값	8
VS	차량 속도 값	8
R_T_S	표준 토크 비율 값	8

표 11. TCU2 CAN 신호
Table 11. TCU2 CAN signal.

신호	신호 설명	비트길이
ETL_TCU	TCU 요청 엔진 토크 최대치	8
TQI_TCU_J	TCU의 토크 개입, 지정된 엔진 토크를 참조	8
Free	Free	8
Free	Free	8
Free	Free	8
Free	Free	8
Free	Free	8

표 12. EMS CAN 신호의 압축 효율 비교
Table 12. Comparison of EMS CAN compression efficiencies.

압축 알고리즘	정상주행		과속/급정거	
	전송 비트 수	압축 효율	전송 비트 수	압축 효율
원 메시지	4,026,880	0%	260,352	0%
능동형 방법	1,456,904	63.82%	98,128	62.3%
재배열 방법	1,717,040	57.36%	117,848	54.74%
제안한 방법	937,248	76.73%	70,048	73.09%

표 13. TCU2 CAN 신호의 압축 효율 비교
Table 13. Comparison of TCU2 CAN compression efficiencies.

압축 알고리즘	정상주행		과속/급정거	
	전송 비트 수	압축 효율	전송 비트 수	압축 효율
원 메시지	4,927,040	0%	260,160	0%
능동형 방법	2,622,416	46.78%	259,088	0.41%
재배열 방법	2,439,240	50.49%	260,160	0%
제안한 방법	1,170,912	76.24%	123,728	52.44%

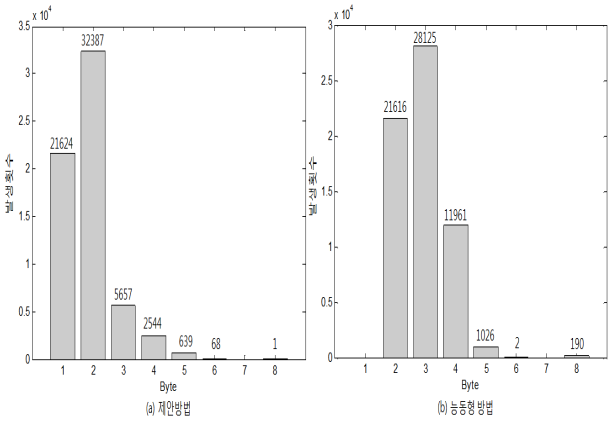

그림 5. 정상주행 시 EMS CAN 데이터 압축 효율 비교.
Fig. 5. Comparison of EMS CAN data compression efficencies under normal driving.

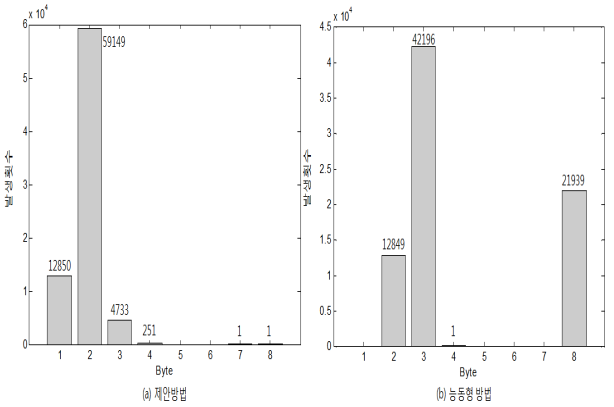

그림 6. 정상주행 시 TCU2 CAN 데이터 압축 효율 비교.
Fig. 6. Comparison of TCU2 CAN data compression efficencies under normal driving.

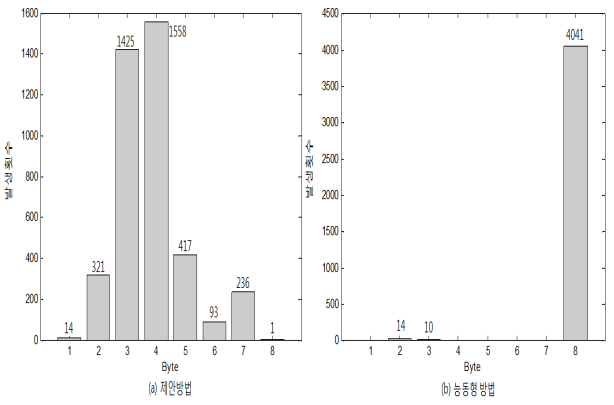

그림 7. 과속/급정거 시 TCU2 CAN 데이터 압축 효율 비교.
Fig. 7. Comparison of TCU2 CAN data compression efficencies under sudden stop/acceleration driving.

2. 임베디드 시스템을 이용한 구현

그림 8은 테스트 환경 블록도이다. 그림 8에서 CANPro Analyzer는 고속 병렬 버스형 USB 인터페이스이며 40MHz 동작 클럭의 DSP MCU에 내장된 CAN 주변 장치를 사용한다. 그림 9는 Coretex M3 임베디드 테스트 보드이다. 테스트 보드는 real-time 애플리케이션을 위한 이더넷 연결을 지원하면서 IEEE1588 Precise Time Protocol를 지원하는 하드웨어와 함께 10/100 Ethernet MAC 기능이 추가되어있다.

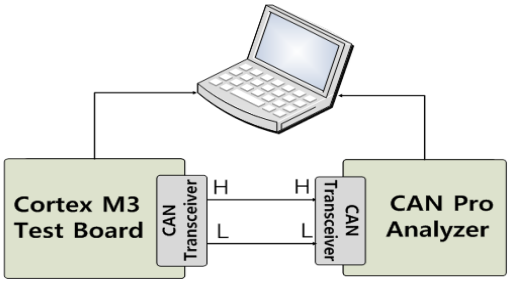

그림 8. 테스트 환경.
Fig. 8. Test environments.

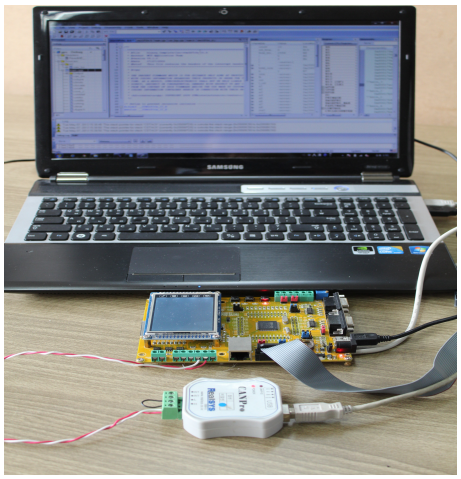

그림 9. Cortex M3 임베디드 테스트 보드.
Fig. 9. Cortex M3 embedded test board.

CAN2.0B	Data Frame	길이:8	ID:00000316	데이터:05 15 58 0C 15 0D 00 00
CAN2.0B	Data Frame	길이:1	ID:00000316	데이터:00
CAN2.0B	Data Frame	길이:2	ID:00000316	데이터:04 05
CAN2.0B	Data Frame	길이:1	ID:00000316	데이터:00
CAN2.0B	Data Frame	길이:2	ID:00000316	데이터:04 09
CAN2.0B	Data Frame	길이:2	ID:00000316	데이터:04 08
CAN2.0B	Data Frame	길이:1	ID:00000316	데이터:00
CAN2.0B	Data Frame	길이:2	ID:00000316	데이터:04 08
CAN2.0B	Data Frame	길이:1	ID:00000316	데이터:00
CAN2.0B	Data Frame	길이:2	ID:00000316	데이터:04 05
CAN2.0B	Data Frame	길이:1	ID:00000316	데이터:00
CAN2.0B	Data Frame	길이:2	ID:00000316	데이터:04 05
CAN2.0B	Data Frame	길이:1	ID:00000316	데이터:00
CAN2.0B	Data Frame	길이:2	ID:00000316	데이터:04 04
CAN2.0B	Data Frame	길이:1	ID:00000316	데이터:00

그림 10. CANPro Analyzer를 이용한 제안한 방법의 시뮬레이션 결과.
Fig. 10. Simulation result by using CANPro Analyzer.

ECU 프로세서는 32비트 마이크로 프로세서 (ARM Cortex_M3)이며 72MHz 주파수에서 고성능 32비트 RISC 코어를 운영할 수 있고 CAN 버스 transceiver와 stand-alone CAN 컨트롤러를 포함한다. 테스트 보드 구동을 위해 IAR workbench툴을 이용하여 C언어로 압축 시스템을 설계 하였다. 테스트 보드와 통신을 위해 CAN Pro Analyzer를 이용하였다. 그림 10은 CANPro Analyzer를 이용한 제안한 방법의 시뮬레이션 결과이다.

시뮬레이션 결과에서 데이터 길이가 8byte인 ID_316 데이터는 처음 보낸 8byte 데이터 이외에 주로 1또는 2byte로 압축된 것을 확인할 수 있다. 한 개의 64비트 EMS CAN 데이터를 압축하는데 0.16ms가 소요된다. EMS CAN 데이터는 10ms 주기로 전송되므로 압축하는데 소요되는 시간은 EMS CAN 데이터의 송수신에 지장을 초래하지 않는다.

V. 결 론

본 논문에서는 DLC와 전송 데이터 압축영역 설정 절차를 사용하여 CAN 메시지 압축 알고리즘을 제안하였다. 제안한 방법에서는 기존의 알고리즘과 달리 변화량을 저장하기 위한 최대 변화량의 범위를 설정하지 않아도 되기 때문에 부정확한 설정에서 발생하는 오류나 지나친 설정에서 발생하는 압축효율저하를 피할 수 있다. 또한, DLC 크기에 의해 압축 유무를 판단함으로써 기존 방법에서와 같이 두 개의 ID로 압축 여부를 판단하는 비효율적인 문제점을 해결할 수 있다. 또한 제안한 방법에서는 특정회사의 CAN 기술표준과 무관하게 데이터 압축이 가능함을 보였다.

정상주행 시와 과속/급정거 시의 차량 CAN 로깅파일로 시뮬레이션 해본 결과 제안한 방법으로 EMS CAN 데이터를 압축하면 77%의 압축률을 얻을 수 있고, TCU2의 경우 과속/급정거 시 능동형 방법에 비해 52%의 추가 압축률을 얻을 수 있음을 보였다.

마지막으로 제안한 CAN 압축 알고리즘을 검증하기 위해 Cortex M3 임베디드 테스트 보드를 이용하여 테스트 했을 때 한 개의 64비트 EMS CAN 데이터를 압

축하는데 0.16ms가 소요되어 실제 차량 응용에 사용가능함을 보였다.

REFERENCES

[1] Road Vehicles-Low Speed Serial Data Communication Part 2: Low Speed Controller Area Network, ISO 11519-2, 1994.
[2] Road Vehicles-Interchange of Digital Information -Controller Area Network for High-Speed Communication, *ISO* 11898, 1994.
[3] Rebert Bosch GmbH, "CAN specification 2.0", *Chuck Powers, Motorola MCTG Multiplex*, April 5, 1995.
[4] S. Channon and P. Miller, "The requirement of future in-vehicle networks and an example implementation", *SAE paper* 2004-01-0206, Mar. 2004.
[5] Wolfhard Lawrenz, *CAN System Engineering: from Theory to Practical Applications.* Sringer, 1997.
[6] Yongwook Son, Heeseok Moon, Jaeil Jeong, and Sooyong Lee, "Active CAN data reduction algorithm for in-vehicle network," *Proceedings of The Korean Society of Automotive Engineers*, pp. 1427-1477, Korea, 2006.
[7] Kyungju Cho, "Variable length can message compression using bit rearrangement," *IEEK of Korea,* 48(4) pp. 51-56, 2011.

저 자 소 개

오 유 경(학생회원)
2011년 중국 연변대학교 정보통신학과 학사 졸업.
2013년 전북대학교 전자공학부 석사 졸업.
2013년~현재 전북대학교 전자공학부 박사과정.
<주관심분야 : 통신, 신호처리, 반도체>

정 진 균(정회원)
1985년 전북대학교 전자공학 학사 졸업.
1989년 미국 미네소타 주립대학 전기공학 석사 졸업.
1991년 미국 미네소타 주립대학 전기공학 박사 졸업.
<주관심분야 : 통신, 컴퓨터, 신호처리, 반도체>