차량차체 네트워크에서 CAN 신호 게이트웨이 설계

A CAN Signal Gateway Design for Car Body Networks

한 준 수, 강 기 호*

(Jun-Soo Han and Ki-Ho Kang)

Abstract: The automobile networks consist of the communication bus systems which become independent and heterogeneous each other. Most often, the CAN buses are implemented in a car in order to connect all the electronic control units in various ways. Thus, some gateways are necessary for exchanging the useful information between electronic control units on the buses. The automobile body networks group is divided into two kinds on a large scale, namely the low-speed bus and the high-speed bus. To interchange messages between the two, a CAN signal gateway is designed and implemented in a miniature scale. A network analyzer (called "Vehicle spy") and an oscilloscope monitor network situation to confirm the due operation of CAN signal gateway. The efficiency of the designed gateway is evaluated. The more message thread increased, the more efficiency decreased.

Keywords: CAN (Controller Area Network), ECU (Electronic Control Unit), gateway, body network

I. 서론

현재의 주요 자동차 기술들은 분산 시스템에서 네트워크 시스템으로 발전해왔다. 다양한 차량 전자 시스템은 다른 네트워크 기술을 사용하는데, 그 중에 CAN 프로토콜이 여러 네트워크 시스템의 중심이 되었다.

자동차 네트워크 시스템은 서로 다른 통신 속도의 CAN 버스 클래스가 존재한다. 파워 윈도우, 좌석 조절 장치, 계기 같은 클래스 B 네트워크들은 10[kbit/s]에서 125[kbit/s]의 중저속인 반면, 파워 트레인, 안전성 제어, 엔진 관리, 변속 같은 실시간 제어를 요하는 클래스 C 네트워크에서는 125[kbit/s] 부터 최대 1[Mbit/s]로 작동한다[1].

이렇듯 개별 차량 시스템에서는 다양한 통신 속도 요구와 호환되지 않는 물리 계층, 보안 유지, 불필요한 통신량을 줄일 수 있도록 버스 분리를 통해 필요한 최소의 정보 공유가 가능하도록 독립된 네트워크로 구성된다[2].

그러나 자동차의 다른 클래스의 버스 사이에서 모든 시스템이 요구하는 정보를 호환되는 형태로 변환하여 적시에 교환하기 위해서는 게이트웨이가 필요하다. 예를 들면, 차량 속도 신호는 엔진 제어 시스템뿐 만 아니라 운전자에게 알리기 위한 계기판으로도 전송되어야 하는데 중간에 실내 계기판 모듈(IPM)이 이러한 역할을 한다[5].

본 논문에서는 클래스 B와 클래스 C를 대표하는 자동차의 인테리어와 파워트레인 노드를 선정하여 두 버스 사이의 정보를 변환하여 교환하는 게이트웨이의 설계 과정을 기술하고 이에 대한 동작 및 메시지의 시간영역 분석을 하였다.

II. CAN 신호 게이트웨이

현재의 상용차에서는 표준화된 게이트웨이 규격은 제조사별 자체 표준을 개발하고 있는 실정이다. 본 논문에서는 특정 제조사에 한정하지 않고 실제 게이트웨이의 기능으로 충족되어야 할 기본 요구항목에 주안점을 두었다.

1. CAN 신호 게이트웨이의 개요

1.1 CAN 신호 게이트웨이의 출현 배경

1986년 독일의 로버트 보쉬에 의해 CAN이 개발된 이후에 다양한 버스 시스템들은 다음의 이유들로 쓰이고 있었다.

- 네트워크 시스템간의 다른 통신 속도차이를 해결하고자 하는 목적
- 통신 버스의 물리 계층간의 호환되지 않는 특성
- 물리적인 분리로 보안의 유지
- 버스 수의 증가를 통해 불필요한 통신 트래픽을 분산시켜 적은 정보 공유하는 시스템의 설계

이렇듯 개별 버스 시스템의 호환성과 트래픽 과다 등의 문제를 해결하기 위해 CAN 네트워크 기반의 멀티플렉스 통신방식의 게이트웨이 시스템이 설계되었다[3].

1.2 CAN 클래스의 구분

CAN 네트워크는 3개의 클래스로 구분된다. 클래스 A 네트워크들은 전자 부트 작동기, 전동 거울 조정장치, 비 탑지, 쉐 루프, 기상 관리 등과 같은 편의 기능 또는 고급 기능들에 일반적으로 사용되며 10[kbit/s] 미만으로 작동한다. 클래스 B 네트워크들은 FAM (Front Area Module), IPM (Instrument Panel Module), DDM (Driver Door Module), ADM (Assist Door Module), RAM (Rear Area Module), Cluster와 같은 정보의 일반적인 전송을 담당하고 10[kbit/s]에서 125[kbit/s] 정도로 작동한다. 클래스 C 네트워크들은 PCU (Powertrain Control Unit), EPS (Electronic Power Steering), ECS (Electronic Control Suspension), ABS (Anti-lock Brake System), CVE (Central Vehicle Electronics)와 같은 정보의 빠른 응답 시간 또는 전송이 필요한 모든 응용에서 사용되고 125[kbit/s]부터 최대 1[Mbit/s]로 작동한다[6].

* 책임저자(Corresponding Author)

논문접수: 2010. 3. 15., 수정: 2010. 4. 15., 채택확정: 2010. 4. 30.

한준수: 한국기술교육대학교 대학원(jshan77@kut.ac.kr)

강기호: 한국기술교육대학교 메카트로닉스공학부(khkang@kut.ac.kr)

* 본 논문은 한국기술교육대학교 RIC의 부분지원 하에 이루어졌음.

그림 1. 자동차 네트워크 클래스B, 클래스C 구분도.

Fig. 1. Diagram for car network classes(Class A,B).

그림 2. 자동차 네트워크 시스템의 구성.

Fig. 2. Configuration of car body network.

2. CAN 신호 게이트웨이의 목적

차량의 IPM (Instrument Panel Module)이 게이트웨이로써 다른 네트워크 클래스의 버스 시스템 사이에서 정보의 형태를 변환하여 표현한다. 게이트웨이의 단순한 목적은 한쪽 버스에서 프레임을 수신하여 정보를 요구하는 상대 버스로 전부 재전송하는 것이다. 또한 엔진 파라메터나 펌웨어 개량과 같은 자동차 시스템 변경을 위한 진단용 통신으로도 사용된다[7].

본 논문에서는 차량 내의 네트워크에서 중·고속의 CAN 통신 버스인 CAN 클래스 B와 클래스 C에 해당하는 전자 제어 유닛의 메시지 교환을 하도록 한다.

III. CAN 신호 게이트웨이의 설계 및 구현

본 논문에서 다룬 게이트웨이 시스템에서는 자동차 환경에서 쓰이는 CAN 네트워크 클래스 B, 클래스 C 시스템의 일부를 CAN 노드로 구현하였다. CAN 게이트웨이 시스템은 모두 6개의 기능 모듈로 구성되어 100[kbps]의 저속 CAN 버스에 연결된 노드 1 블록과 500[kbps]의 고속 CAN 버스에 연결된 노드 2 블록이 게이트웨이를 통해 서로 연결된 구성을 가지고 있다.

저속과 고속의 CAN 데이터 관리자라는 중간 관리자를 통해서 노드 간의 메시지 전송 요구와 데이터 변경을 주관하게 된다.

그림 3. 게이트웨이 시스템의 구성.

Fig. 3. Configuration of gateway system.

1. 하드웨어의 구성

게이트웨이는 모토롤라사의 S12 코어 기반의 2개의 CAN 채널을 구비한 16비트 마이크로컨트롤러인 MC9S12DG128 [10]을 사용하였다. 보드의 구성으로는 설정 상태를 표시해 주는 LCD, 디버깅을 위한 시리얼 포트, 저속 및 고속 CAN 버스 접속을 위한 9핀 DSub 포트, 네트워크 분석기와의 통신을 위한 25핀 DSub 포트가 있다.

노드는 마이크로칩사의 PIC18 계열의 1개의 CAN 컨트롤러를 내장한 8비트 마이크로컨트롤러인 PIC18F248, PIC18F6680 [11]을 사용하였다. 액추에이터(Actuator)로는 실제 차량에서 사용되는 방향 지시등, 도어 락, 윈도우 모터, 부저를 선정하였고, 입력 신호로는 다기능 스위치와 충돌 감지 센서 입력을 대신하는 스위치로 구성한다.

게이트웨이 시스템의 노드들은 6개의 입력과 액추에이터 구동을 위한 6개의 출력으로 구성된다. 시스템 구성 시 노드 1은 클래스 B, 노드 2는 클래스 C로 정한다. 또한 CAN 메시지 ID는 우선 순위 및 네트워크 클래스를 고려하여 고

표 1. 게이트웨이 시스템의 입·출력 구성.

Table 1. I/O configuration of gateway system.

모듈	ID	입력	출력	CAN 메시지
Node 1	0x0100	LAMP IN	BUZZER OUT	LS NODE1 STATUS - LAMP STATE - WINDOW STATE
	0x0200	WINDOW UP IN	LOCK OUT	
		WINDOW DN IN	UNLOCK OUT	
Node 2	0x0100	LOCK IN	WINDOW UP OUT	HS NODE2 STATUS - LOCK STATE - CRASH STATE - LOCK COMMAND
		UNLOCK IN	WINDOW DN OUT	
	0x0010	CRASH IN	LAMP OUT	
Node 3	0x0700	LS NODE1 STATE		- LAMP - WINDOW
	0x0300	HS NODE2 STATE		- LOCK - CRASH

그림 4. 버스 인터페이스 회로도.

Fig. 4. Schematic diagram for interface circuits.

사진 1. 게이트웨이 데모 보드.

Photo1. Demo board of gateway.

속 CAN 버스에 위치한 노드 2를 낮은 번호로 하고, 게이트웨이는 큰 ID 번호를 할당하였다. 그러나 실제 차량에서의 노드별 부하 그룹핑 기준을 맞추기보다는 구현상의 특성을 반영하여 아래와 같이 구성하였다.

다음은 CAN 게이트웨이와 네트워크 분석기, 노드와 액추에이터를 결합한 데모 보드이다. 버스는 이상적인 조건하에 신호 전달 지연 요소를 고려하지 않고 10[m] 이내로 구성하였다.

2. 프로그램의 구성

게이트웨이 프로그램은 게이트웨이 모듈 API, 게이트웨이 모듈 소스, 저속 CAN 신호 데이터 관리 모듈 API, 저속 CAN 신호 데이터 관리 모듈 소스, 고속 CAN 신호 데이터 관리 모듈 API, 고속 CAN 신호 데이터 관리 모듈 소스로 구성된다.

게이트웨이 모듈 프로그램은 CAN 메시지의 전송 주기를

그림 5. 게이트웨이 프로그램의 구성.

Fig. 5. Configuration for gateway program.

100[ms]단위로 설정하고, 게이트웨이 모듈을 초기화 및 게이트웨이 모듈의 동작을 위한 함수를 정의한다. CAN 신호 데이터 관리 모듈 프로그램은 해당 노드의 액추에이터 모듈별 CAN 데이터가 변경되었을 때 실행될 함수의 타입을 선언하고 저속이나 고속의 CAN 버스에 연결된 모듈과 신호의 수를 정의한다. 또한 CAN 신호의 수에 따른 로컬 버퍼의 용량, 노드별 CAN 신호의 ID와 DLC, 게이트웨이의 CAN 신호 ID와 DLC를 정의한다. 노드별 CAN 모듈의 초기화, 노드와 게이트웨이의 CAN 메시지가 변경되었을 때 변수값을 갱신하기 위한 각각의 감지자 함수, 해당 노드의 액추에이터 모듈과 게이트웨이의 메시지의 변경 확인, 외부에서 받은 노드 모듈의 CAN 메시지를 확인하고 로컬 데이터를 변경하는 과정을 거친다.

CAN 클래스 B와 클래스 C 중에서 데모 시스템의 노드별 통신 속도로 선택한 CAN 비트 타이밍이다[12]. 공통 사

그림 6. 노드1 비트 타이밍도.

Fig. 6. Timing diagram for node 1.

그림 7. 노드2 비트 타이밍도.

Fig. 7. Timing diagram for node 2.

표 2. 게이트웨이 메시지 ID의 구성.

Table 2. Composition of gateway message ID.

Priority	Reserved	Data page	PDU format	PDU specific	Source address
			PGN		
3bit	1bit	1bit	8bit	8bit	8bit

향으로는 $F_{osc} = 16[\text{Mhz}]$ ($T_{osc} = 62.5[\text{ns}]$), $\text{SyncSeg} = 1$ TQ, $\text{PropSeg} = 1$ TQ, $\text{PhaseSeg1} = 3$ TQ, $\text{PhaseSeg2} = 3$ TQ, $\text{SJW} = 1$ TQ이다.

게이트웨이에서 메시지 ID는 두 개의 노드에 대해 메시지 전송을 피어 투 피어(peer-to-peer) 네트워크 방식으로 정의한다.

본 연구에서는 J1939 [7,8]에서 제안된 목적지 주소, 데이터 페이지 비트, 우선순위, 소스 주소, 파라미터 그룹 번호 (PGN: Parameter Group Number)를 묶어 게이트웨이 메시지 ID로 정의하였다.

최대 8 바이트까지의 파라미터 그룹을 CAN 메시지로 전송하고 PGN을 통해 유일한 ID 주소가 부여된다. 16비트 값으로 프로토콜 데이터 유닛(PDU: Protocol Data Unit)의 포맷과 상세를 구성한다.

3. 게이트웨이 메시지 처리 알고리즘

초기의 중요한 인터럽트 처리로는 MSCAN [13] Rx interrupt, MSCAN Tx buffer empty interrupt, Gateway timing interrupt [14]로서 이를 기반으로 한 게이트웨이 알고리즘이다.

3.1 ID의 검색

2진 검색 알고리즘이 배열 요소 정렬의 기반으로 하여 Rx 프레임 ID를 오름차순으로 정렬한다.

2진검색은 전체 Rx 프레임 ID를 두 개의 서브배열로 분리해가면서 Rx 프레임 서술자를 검색하는 방식으로 찾고자 하는 Rx 프레임 서술자 값을 중간 배열 값과 비교하면서 검색하는 방식이다.

표 3. 게이트웨이 메시지 ID PGN의 정의.

Table 3. PGN definition of gateway message ID.

Data Length		8 Byte	8 Byte
Data Page		0	0
PDU Format		7	3
PDU Specific		00	00
Default Priority		5	4
Parameter Group Number		1792 (0x700)	768 (0x300)
Start Position	Length	Parameter Name	
0 [0..1]	2 bit	LAMP	LOCK
0 [2..3]	2 bit	-	UNLOCK
0 [4..5]	2 bit	WINDOW DOWN	-
0 [6..7]	2 bit	WINDOW UP	-
1 [0..7]		-	-
7 [0..1]	2 bit	-	CRASH
7 [2..7]		-	-

```

tRxFrameDescr *RxFindFrameId (int FrameId,
tNodeDescr *node) {
    tRxFrameDescr *base = (node -> rx_idx_start_p);
    int i, result;
    tRxFrameDescr *p;
    for(i = node->rx_idx_cnt; i != 0; i >>= 1){
        p = base + (i >> 1);
        result = FrameId - (p->ID);
        if (result == 0) return(p);
        if (result > 0) {
            base = p + 1;
            i--;
        }
    }
    return (NULL);
}

```

중간 레코드 번호(m) 다음과 같다. (l : 첫번째 Rx 프레임 서술자 번호, u : 마지막 Rx 프레임 서술자 번호)

$$m = \frac{(l+u)}{2}$$

오름차순으로 배열된 Rx 프레임 서술자 배열에서 중간 Rx 프레임 서술자(K_n)와 검색하고자 하는 Rx 프레임 서술자(R)를 비교한 결과에 의해 찾아가는 검색 방법이다.

$R < K_n$ 이면 키값은 K_n 보다 작은쪽 파일에 있고, $R = K_n$ 이면 중간 레코드가 찾는 레코드이다.

$R > K_n$ 이면 키값은 K_n 보다 큰쪽 파일에 있고, $R = K_n$ 일 때 검색은 끝난다.

일치된 Rx 프레임 서술자에 대해 포인터 값으로 ID를 반환하고 일치하지 않는 ID가 있는 경우에는 Rx 프레임 서술자에 대해 Null 포인터 값을 부여한다. 검색의 비교 횟수는 $\log_2 n + 1$ 로 많은 Rx프레임 서술자 테이블의 검색에서 특정한 노드의 위치를 검색할 때 효율적이다.

3.2 CAN 프레임 수신

수신된 프레임에서 정보를 찾기 위한 처음 단계는 Rx 프레임 서술자에 대한 게이트웨이 데이터베이스를 찾는 것이다. 데이터베이스에 프레임에 대한 정보가 없다면, 더 이상 진행하지 않고 실행을 마친다. 다음으로 Rx 프레임 서술자에서 일치하는 ID가 발견되면, 수신된 프레임의 데이터 크기를 검사한다. 프레임이 예상된 바이트의 수보다 적다면 무효 프레임 처리 루틴이 호출되고 더 이상 진행되지 않는다. 데이터 크기 검사를 한 후에 게이트웨이는 Rx 타이머를 다시 추가한다. 프레임에 대한 실패 지시자가 활성화되면 그것을 소거시키고 데이터는 개별 신호로 나뉘어 목적지로 복사된다.

그림 8. CAN 프레임 수신 알고리즘.

Fig. 8. Receive algorithm of CAN frame.

3.3 CAN 프레임 전송

프레임 전송은 전송 대기 열 스케줄링에 의해 진행되고 포인터를 적절한 대기 열의 프레임 서술자에 삽입한다. 루틴이 실행될 때, 전송 예약 플래그가 한번 이상 전송 대기 열로 들어갈 수 없도록 하기 위해 전송 예약된 프레임인지를 확인 한다. 전송이 되면, 신호는 최신의 이용 가능한 값을 포함하게 된다.

예약된 프레임으로 표시된 후에는, Tx 프레임 서술자의 포인터를 적절한 노드의 전송 대기 열의 버퍼로 삽입한다. 버퍼의 크기는 모든 노드의 프레임을 수용할 수 있도록 충분한 공간으로 정하고, 버퍼 대기 열이 과다가 되지 않도록 한번에 모든 프레임의 전송 일정을 관리한다.

주기적인 전송일 경우에는 새로운 주기를 시작하는 Tx 타이머를 다시 시작한다. 프레임이 전송 예약이 되거나 실제로는 전송이 되지 않아도 타이머는 다시 시작한다. 현재 주변 장치가 유휴 상태이면, 최종적으로 전송이 이루어진다. 전송할 프레임이 더 이상 없을 때, 인터럽트를 해제한다.

3.4 Rx와 Tx 타이머 취급

게이트웨이의 부하를 줄이기 위해, 인터럽트 신호에 대해 마이크로 프로세서의 하드웨어 타이머는 일정한 10[ms] 주기로 실행한다.

타이머의 프리스케일러 값은 게이트웨이가 요구하는 일정한 전송 주파수와 사용자가 관독하기 쉬운 수신 종료 시간을 기준으로 설정한다. 보통의 경우에는 타이머가 실행되는 기간에는 타이머 값이 0 이 되지 않고 타이머 값이 감소하여 0 이 되면 만기가 되어 개별 Tx 타이머, Rx 타이머를 실행한다.

그림 9. CAN 프레임 송신 알고리즘.

Fig. 9. Transmit algorithm of CAN frame.

3.5 게이트웨이 메시지 동작 및 측정

HS INT STATUS 메시지는 저속 CAN 버스에서 받은 인터리어부의 CAN 상태정보를 고속 CAN 버스로 보낸다. 노트 1의 LAMP 신호를 고속 CAN 버스로 전송하면, 게이트웨이에서는 인터리어 블록의 상태를 나타내는 CAN 메시지 데이터 프레임에 해당 정보를 2비트로 나타낸다. (00=INVALID, 01=LAMP OFF, 10=LAMP ON) 또한 윈도우의 모터 구동을 위한 업·다운 신호는 각각 2비트로 구성

그림 10. 타이머 처리 송·수신 알고리즘.

Fig. 10. Timer-based T/R algorithm.

그림 11. HS INT STATUS 신호 변화에 따른 파워윈도우 동작 타이밍도.

Fig. 11. Action timing of power window by signal change in HS INT STATUS.

그림 12. HS INT STATUS 신호 파형 전·후 변화.

Fig. 12. Change of before/after HS INT STATUS signals.

그림 13. LS PT STATUS 신호 변화에 따른 도어 락 동작 타이밍도.

Fig. 13. Action timing of door lock by signal change in LS PT STATUS.

그림 14. LS PT STATUS 신호 파형 전·후 변화.

Fig. 14. Change of before/after LS PT STATUS signals.

된다. (00=INVALID, 01=WINDOW UP&DOWN ON, 10=WINDOW UP&DOWN OFF) 노트 1의 비상 램프 스위치를 켜고 노트 2의 윈도우 모터를 다운 동작을 하도록 하는 게이트웨이의 일련의 처리 절차를 도시하였으며, 고속 CAN 버스로 전송된 게이트웨이 신호 변화를 측정하였다.

Filter	Count	Time	Tr	Er	Description	Arbitrator	DataBytes	Network	Node
CRASH	101.000 ms				AS_STATE	10	01	NORMAL [1]	HS CAN
WIPER	100.008 ms				BCM_STATUS	100	02 00	WIPER_OFF [2]	LSFT CAN
DS_WINDOW_STATE	101.039 ms				DDM_STATUS	200	05 12 05	IDLE [1]	LSFT CAN
DS_WINDOW_POS								LOCK [1]	
AS_WINDOW_CMD								18 [12]	
AS_LOCK_CMD								IDLE [1]	
ECU_STATE	11.020 ms				ECU_STATE	100	C8 00	IDLE [1]	HS CAN
SPEED	101.011 ms				INT_STATUS	700	92 00 00 00 00 00 00 00	200 km/h [08]	HS CAN
WIPER								WIPER_OFF [2]	
LOCK								LOCK_ON [1]	
UNLOCK								UNLOCK_OFF [2]	
PT_STATUS	100.963 ms				PT_STATUS	300	C8 00 00 00 00 00 00 01	200 km/h [08]	LSFT CAN
CRASH								NORMAL [1]	

그림 15. Vehicle Spy를 통한 통신 모니터링.

Fig. 15. Comtion monitoring through “Vehicle Spy”.

LS PT STATUS 메시지는 고속 CAN 버스에서 받은 파워트레인부의 CAN 상태정보를 저속 CAN 버스로 보낸다. 노드 2의 CRASH 신호를 저속 CAN 버스로 전송할 경우, 게이트웨이에서는 파워트레인 블록의 상태를 나타내는 CAN 메시지의 데이터 프레임에 해당 정보를 2비트로 나타낸다. (00=INVALID, 01=NORMAL, 10=CRASH DETECTED) 또한 도어 락 모터를 구동하기 위한 닫힘, 열림 신호는 각각 2비트로 구성된다. (00=INVALID, 01=LOCK&UNLOCK ON, 10=LOCK& UNLOCK OFF) 노드 2로부터 충돌 신호가 입력되었을 때 노드 1의 도어 락 모터를 동작하도록 하는 게이트웨이의 일련의 처리 절차를 도식하였으며, 저속 CAN 버스로 전송된 게이트웨이 신호를 측정하였다.

4. 게이트웨이 메시지 시뮬레이션

다음은 게이트웨이의 메시지 전송 상황을 네트워크 분석기와 PC상의 Vehicle Spy[9]와의 통신을 통해 모니터링 하도록 노드의 기능을 자바를 사용하여 시각적으로 구현하고, 노드별 액츄에이터 동작 시 메시지 전송 상태를 시뮬레이션한 결과이다.

IV. 제안된 게이트웨이의 시간영역 분석

게이트웨이의 시간영역 측정을 위한 도구로 Metrowerks사의 컴파일러 Codewarrior V5.1을 사용하였다. 마이크로컨트롤러에서 요구되는 메모리 크기를 측정하고, CPU에 내장된 램에 간헐적으로 접근할 때 게이트웨이의 신호 복사 실행 시간을 측정하였다.

1. CAN Rx 쓰레드

노드들로부터 개별 CAN 메시지를 수신한 후 Rx 쓰레드

표 4. 시간영역 평가 제한 조건.

Table 4. Constraints for time-domain appraisal.

파라미터	허용 범위
Frame data size	0 - 15 bytes
Signal size	1- 64 bits
Signal position	0 - 63
Number of Tx buffers	0 - 1024
Rx / Tx timer period	1 - 5120 ticks (51.2s max. @ 10ms time base)
Frame ID length	1 - 11 bits

가 실행된다. ID 탐색, 신호 복사, 프레임 전송 스케줄링을 분석하였다.

1.1 ID 탐색

수신된 메시지에 대해 ID가 일치하는 Rx 프레임 서술자를 찾는 ID 탐색 시간을 측정한다. 탐색 시간은 탐색 알고리즘에 의해 요구되는 반복 수와 거의 일치하는 추세이다. Rx 서술자 테이블의 크기를 최대 200까지 했을 때, 마이크로컨트롤러 CPU의 버스 사이클을 측정한 결과이다.

1.2 신호 복사

수신된 프레임의 크기를 검증하고 신호가 목적지 버퍼로 복사되는 과정은 주요한 처리 작업이다. 데이터 크기별로 비교했을 때 바이트 크기보다 비트 정확도로 데이터 복사가 진행될 때 더 많은 버스 사이클을 요구한다.

1.3 프레임 전송 스케줄링

전송 스케줄링 알고리즘에 의해 특정 프레임의 전송 예정 여부가 처리 실행 시간을 결정하는데 전송이 예정된 프레임일수록 실행 시간이 적게 소요됨을 확인하였다.

2. CAN Tx 쓰레드

MSCAN Tx 버퍼 비움 인터럽트 요청에 의해 Tx 쓰레드가 실행된다. 인터럽트 서비스 루틴의 실행 시간은 전송 대기열이 예정되었을 때가 비워졌을 때보다 더 많이 요구된다.

그림 16. Rx 서술자 테이블의 사이즈별 최대 ID탐색 시간.

Fig. 16. Maximum ID searching time for the size of Rx descriptor table.

표 5. 프레임 전송 예정 상태에 따른 실행 시간.

Table 5. Execution time for previewed frame transmit status.

전송 상태	실행 시간 [bus cycles]
기 전송 예정된 프레임	12
전송 미 예정된 프레임	40

표 6. Tx인터럽트 서비스 루틴의 실행 시간.

Table 6. Execution time for Tx ISR.

전송 상태	실행 시간 [bus cycles]
전송 예정된 대기열	53
빈 전송 대기열	12

V. 결론 및 연구 목표

1) 본 연구에서는 두 개의 노드를 연결시켜 자동차의 저속과 고속의 버스가 상호 메시지 교환을 통해 실제 차량의 액츄에이터를 구동하는 게이트웨이의 하드웨어 및 소프트웨어를 설계하고 구현하였다.

2) CAN 네트워크 클래스별 속도 차이를 이용한 ID 우선 순위를 부여한 게이트웨이 메시지 처리를 통해 구분된 버스 간에 응급 신호의 공유 및 해당 기능의 동작이 가능하였다.

3) 네트워크의 시간영역 분석 결과 제안된 게이트웨이 프로그램은 노드 제어기의 CPU 버스 사이클을 줄이고 트래픽 관리에 효과가 있음을 확인하였다[4].

그러나 어느 정도 이상의 노드와 목적지의 수, 신호의 길이, 메시지의 대기 빈도의 증가에서는 게이트웨이의 송·수신 지연을 발생시켜 시스템 성능에 악영향을 줄을 확인하였다.

4) 본 연구는 기본적인 게이트웨이의 요건을 평가하기 위해 컴파일러를 통해서 마이크로프로세서의 버스 사이클을 측정하고 PC와 네트워크 분석기로는 메시지를 정의해 차량 네트워크 환경의 시뮬레이션을 통해 접근하였다. 전체 차량 네트워크의 액츄에이터 모듈 기능의 실제적인 구현과 측정의 어려움을 해결하였다.

5) 본 연구의 게이트웨이가 실제 상용차에 활용되기 위해서는 여러 가지 기능개선 사항이 필요하다. 고속에서 저속 CAN 메시지로 변환시 CPU 오버로드를 감소시킬 방안과 송·수신 쓰레드를 개선하기 위한 메시지 변환 효율의 향상, ID 탐색 시간 최적화, 프레임 스케줄링 기법 등의 연구를 진행할 것이다.

6) 데모 보드 상의 구현에서 버스의 길이와 액츄에이터 선정과 같은 구성의 제약 사항을 개선하여 향후 차량 시스템에 적용된 IP 게이트웨이를 최종적으로는 대체하여 실제 모듈간의 다양한 클래스의 네트워크로 확장할 수 있도록 프로그램을 개발하고 실험을 진행할 것이다.

참고문헌

- [1] CAN Specification, Version 2.0, Parts A and B, Robert Bosch GmbH, 1991.
- [2] K. Etschberger, *Controller Area Network Basics, Protocols, Chips and Applications*, IXXAT Press, 2001.
- [3] D. Paret, *Multiplexed Networks for Embedded Systems*, SAE International, and John Wiley & Sons, 2007.
- [4] 이 석 외, "Dual CAN 네트워크의 Load Balancing을 위한 네트워크 트래픽 예측기 개발," 한국자동차공학회 전기·전자시스템 부문 심포지엄 논문집, pp. 101-106, 2006.
- [5] 박평선 외, "차량용 스마트 센서 네트워크 게이트웨이 구현," 한국자동차공학회 전기·전자시스템 부문 심포지엄 논문집, pp. 52-57, 2007.
- [6] SAE J1850, Class B Data Communication Network Interface, Feb. 1994.
- [7] SAE J1939-31, Surface Vehicle Recommended Practice-Network Layer, Apr. 2004.
- [8] SAE J1939-71, Surface Vehicle Recommended Practice-Vehicle Application Layer, Dec. 2004.
- [9] "Vehicle Spy Documentation," Intrepid Control Systems inc., May. 2006.
- [10] MC9S12DT128DGV2 Data Sheet, Freescale Semiconductor Inc., 2005.
- [11] PIC18FXX8 Data Sheet, Microchip Technology Inc., Sep. 2007.
- [12] MCP2510 Data Sheet, Microchip Technology, Inc., 2007.
- [13] S12MSCANV2 Data Sheet, Motorola Inc., July 2004.
- [14] XGATE Library: Signal Gateway, Application Note AN3333, Freescale Semiconductor Inc., Nov. 2006.

한 준 수

2003년 2월 한국기술교육대학교 전자공학과 졸업. 2009년 2월 한국기술교육대학교 대학원 석사. 관심분야는 자동차 전자 및 제어

강 기 호

1985년 2월 서울대학교 전기공학과 졸업. 1987년 2월 서울대학교 대학원 석사. 1996년 8월 INP Grenoble Ph.D(제어공학). 관심분야는 분산 임베디드 제어시스템, 전기자동차.