지능형 자동차의 분산형 시스템을 위한 FlexRay 네트워크 시스템의 구현

Implementation of FlexRay Network System for Distributed Systems of Intelligent Vehicle

하 경 남, 이 원 석, 이 석*, 이 경 창
(Kyoung Nam Ha, Won Seok Lee, Suk Lee, and Kyung Chang Lee)

Abstract : Safety critical systems such as x-by-wire systems require in-vehicle network systems that can interconnect various sensors, actuators, and controllers. These networks need to have high data rate, deterministic operation, and fault tolerance. Recently, FlexRay protocol that is a time-triggered protocol has been introduced, and many automotive companies have been focusing on this protocol. This paper presents a design method of FlexRay network system and implementation of FlexRay-based motor control system.

Keywords : safety critical systems, x-by-wire, in-vehicle network system, FlexRay, time-triggered protocol

I. 서론

최근 들어 자동차 관련 기술의 발전으로 인하여, 차량의 성능 향상과 운전자의 편의성 및 안전성에 대한 요구가 더욱 증가되고 있다. 특히, 차량의 지능화를 목적으로, 기존의 기계적인 장치들과 유압 장치들이 상당 부분 마이크로컨트롤러와 전기 및 전자 장치로 빠르게 바뀌어 가고 있는 추세이다. 예로, 파워 스티어링(Electronic Power Steering, EPS)이나 능동 서스펜션 시스템(Active Suspension System, ASS) 등은 전자 제어 유닛(Electronic Control Unit, ECU)에 의하여 기존의 기계 시스템과 전기 전자 시스템이 통합된 지능형 자동차의 대표적인 시스템들이다.

그러나, 지능형 차량에서 요구되는 다수의 ECU를 일대일로 연결하는 결선 방식은 ECU가 늘어남에 따라 많은 문제점을 일으키고 있다. 즉, 연결되어야 할 ECU가 많아짐에 따라 배선 무게가 증가되거나, 이로 인한 성능 저하 및 연결부의 호환 문제, 그리고 차량의 설계에 있어서 확장성과 설계의 자유도를 제한하는 문제가 그것이다. 이에 따라, 최근에는 차량 전자 시스템에 통신 네트워크를 접목시킨 차량 내 네트워크(In-Vehicle Networking, IVN)가 활발하게 연구되고 있다[1].

뿐만 아니라, 지능형 자동차에 적용되는 전자 제어식 안전성 프로그램(Electronic Stability Program, ESP)이나 순항 제어 시스템(Advanced Cruise Control, ACC)과 같은 첨단 안전 시스템에 대한 관심도 증가되고 있다. 특히, 이러한 첨단 안전 시스템이 증가함에 따라, ECU가 독립적으로 태스크를 수행하고 네트워크를 통하여 그 정보를 공유하는 방식인 분산형 시스템에 대한 연구도 중요하게 다루어지고 있다[1,2].

지능형 자동차에서 차량 내 네트워크와 분산형 시스템이 적용된 예로써, 그림 1과 같은 통합 샤시 제어(Unified Chassis Control, UCC) 시스템을 들 수 있다. 이 시스템에서 각각의 ECU는 기본적으로 편의성과 안전성을 향상시키기 위하여 고유의 제어 기능을 독립적으로 수행함과 동시에, 네트워크를 통하여 그 제어를 위한 정보를 공유하는 형태로 구현된다.

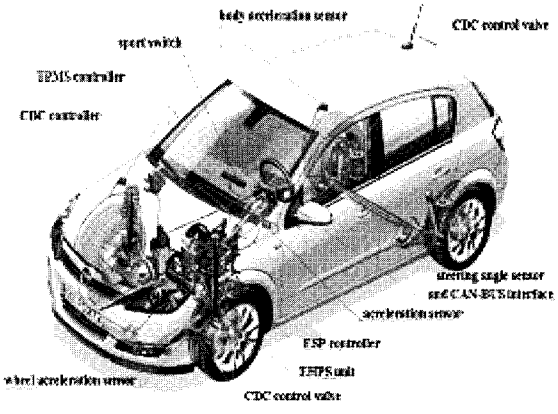


그림 1. 통합 샤시 제어 시스템의 개념도.
Fig. 1. Schematic diagram of unified chassis control system.

현재까지, 다양한 프로토콜이 분산형 시스템을 위한 차량 내 네트워크로 개발되었다. 대표적인 프로토콜로서, 사건 기반 프로토콜(event-triggered protocol) 방식으로 널리 사용되고 있는 CAN(Controller Area Network)과, 시간 기반 프로토콜(time-triggered protocol) 방식인 TTP/C(Time Triggered Protocol/Class C)와 TTCAN(Time Triggered Controller Area Network) 등이 개발되었다. 특히, 최근에 개발된 FlexRay는 고속의 통신 성능과 정시성 보장 및 고장 허용 등을 주요 특징으로 가지고 있다고 알려져 있다[3,4].

본 논문에서는 이러한 FlexRay 프로토콜을 이용하여 지능형 자동차에 적용하기 위한 분산형 네트워크 설계 방법을 제안한다. 또한, 모터 제어 시스템에 FlexRay를 적용하여 구현함으로써 분산형 제어 시스템을 위한 적용 가능성을 검토한다.

본 논문은 서론을 포함하여 5장으로 구성되어 있다. II장에서는 FlexRay 프로토콜의 전반적인 개요를 설명하고, III장에

* 책임저자(Corresponding Author)
논문접수 : 2007. 8. 5., 채택확정 : 2007. 9. 1.
하경남, 이원석 : 부산대학교 지능기계공학과 대학원
(leews@pusan.ac.kr/0vincent@pusan.ac.kr)
이경창 : 부경대학교 전기제어공학부(gclee@pknu.ac.kr)
이 석 : 부산대학교 기계공학부(slee@pusan.ac.kr)

서는 FlexRay 네트워크 설계 방법에 대하여 기술한다. IV장에서는 이러한 설계 방법을 이용하여 실제로 모터 제어 시스템을 구현하여 그 성능을 평가한다. 마지막으로 V장에서는 결론과 향후 과제를 제시한다.

II. FlexRay 의 개요

1. FlexRay 프로토콜의 구조 [5]

FlexRay는 BMW, 다임러크라이슬러, 필립스, 모토로라를 주축으로 하는 FlexRay 컨소시엄에 의해 2000년 이후 개발된 프로토콜이다. 프로토콜의 개발에 있어서, FlexRay는 처음부터 차량 내부의 고속 제어 용도에 적합하며 안전성, 신뢰성, 편의성을 강화한다는 목표를 가지고 추진되었다.

이러한 이유로 FlexRay는 고 대역폭, 내결함성, 정시성을 필수로 하는 전자식 브레이크(brake-by-wire)나 전자식 조향장치(steer-by-wire) 등으로 대표되는 전자화(by-wire) 제어 시스템에 가장 적합한 대안으로 제시되고 있다. 또한 FlexRay 컨소시엄에 의해 첨단 파워트레인, 샤시 시스템을 시작으로 하여 신뢰성이 요구되는 차량 내 통신 시스템의 범세계적인 표준 프로토콜로 추진되고 있다.

FlexRay는 통신 채널(communication channel)당 최대 10Mbps의 데이터 전송 속도를 지원하며 두 통신 채널이 개별적으로 작동하므로 총 20Mbps의 데이터 전송 속도를 가능하게 한다. 시간 기반 프로토콜 방식인 FlexRay는 하나의 통신 사이클 내에서 동기식과 비동기식 프레임 전송이 가능하다. 동기식 전송에서는 프레임 대기 시간과 지터를 보장해 주고, 비동기식 전송에서는 우선 순위에 의한 프레임 전송이 가능하다. 또한, 고장 허용을 위한 2개의 채널 사용이 가능한 특징이 있다.

그림 2는 FlexRay Ver. 2.1의 노드 구조와 각 모듈간의 인터페이스를 나타내고 있다. 그림에서 FlexRay 노드는 1개의 호스트(host)와 1개의 통신 제어기(communication controller), 2개의 버스 드라이버(bus driver)로 구성되어 있고, 각각의 통신 채널은 하나의 버스 드라이버를 가지고 있으므로 독립적으로 동시에 사용이 가능한 구조로 되어 있다.

또한, FlexRay는 가장 기본적인 통신 사이클에 기반하여 동작되는데 자세한 구조는 그림 3과 같다. 이러한 통신 사이클은 정적 구간(static segment), 동적 구간(dynamic segment), 네트워크 유휴 시간(network idle time) 그리고 심볼 윈도우(symbol window)로 구성되어 있다. 이렇게 구성된 통신 사이클을 이용하여, 프레임 데이터가 각각 바이트 단위로 잘라져서 전송된다. 이 때, 정적 구간인 경우 TSS(Transmission Start Sequence), FSS(Frame Start Sequence), BSS(Byte Start Sequence), FES(Frame End Sequence)를 첨가하고 동적 구간인 경우 TSS, FSS, BSS, FES, DTS(Dynamic Trailing Sequence)를 각각 첨가함으로써 보다 유연한 TDMA(Time Division Multiple Access) 전송이 가능하게 된다.

2. FlexRay 프로토콜 전송 메커니즘

FlexRay 프로토콜에서 매체 접근 제어는 앞서 기술한 것과 같이 연속적인 통신 사이클에 기반을 두고 있다. 2가지 매체 접근 방법인 정적 TDMA 방법과 mini-slotting에 기반을 둔 동적 TDMA 방법이 동시에 사용된다.

그림 4는 정적 구간의 구조를 나타내고 있다. 정적 구간 내에 있는 모든 통신 슬롯들은 동일한 길이로 구성되어 있다. 따라서 동기화에 필수적인 싱크 프레임(sync frame)은 이 구간 내에 있어야 하고 2개의 채널로 동시에 전송함으로써 성공적인 동기화를 보장해 줄 수 있어야 한다. 이렇게 정해진 구간 내에서 오직 하나의 노드만이 사전에 스케줄링된 고유한 프레임 ID를 이용해 채널에 접근하여 전송할 수 있는 권한을 가지게 된다.

이와 달리, 그림 5에서 보여주는 동적 구간 내에서는 mini-slotting을 이용하여 가변적인 통신 슬롯의 길이를 지원한다. 이를 이용해서 가변적인 프레임 길이의 데이터를 전송할 수

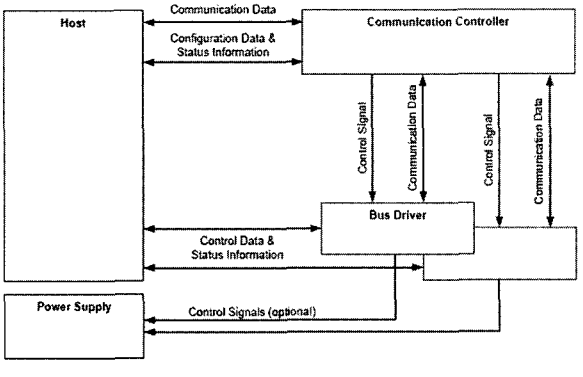


그림 2. 노드 구조와 논리 인터페이스.
Fig. 2. Node architecture and logical interface.

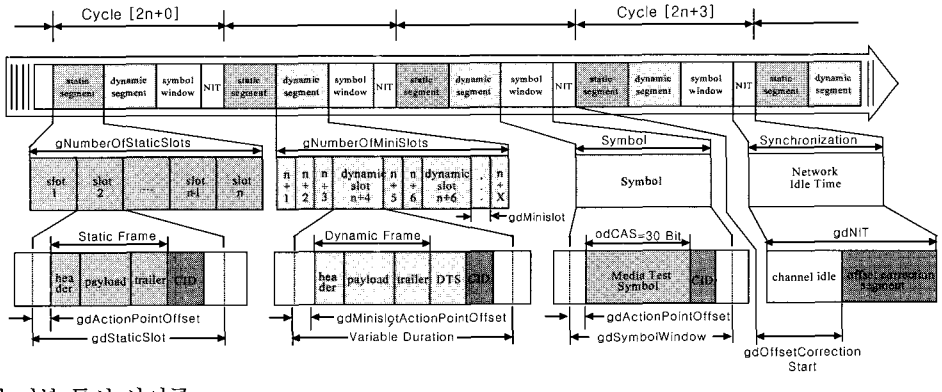


그림 3. FlexRay의 기본 통신 사이클.
Fig. 3. Basic communication cycle of FlexRay.

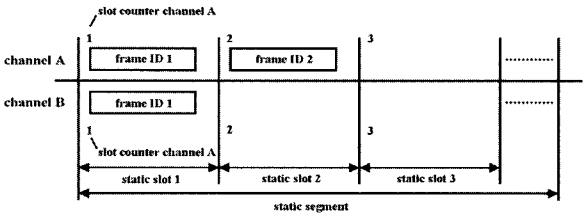


그림 4. 정적 구간의 구조.
Fig. 4. Structure of static segment.

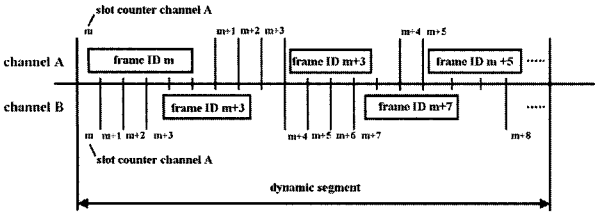


그림 5. 동적 구간의 구조.
Fig. 5. Structure of dynamic segment.

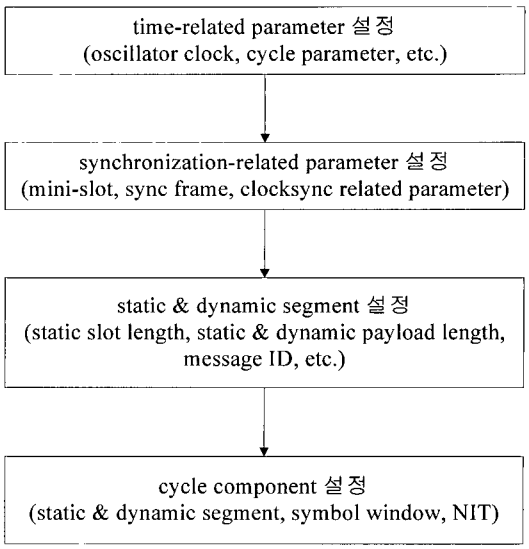


그림 6. 클러스터 설계 절차.
Fig. 6. Procedure of cluster design.

있게 되며 각각의 노드가 전송하고자 하는 데이터를 중재(arbitration) 할 수 있게 된다. FlexRay의 전송 메커니즘의 주요한 특징인 전송 중재 기능은 각각의 노드가 유지하고 있는 2개의 슬롯 카운터(slot counter)를 이용하여 이루어진다. 이 슬롯 카운터는 독립적으로 동적으로 증가하게 되는데, 전송을 하고자 하는 각각의 노드는 동적 구간 내에서 프레임 전송이 시작될 수 있는 마지막 소구간(minislot)의 번호를 확인하여 전송을 시작할 것인지 포기할 것인지를 중재할 수 있게 된다.

FlexRay는 다수결 투표(majority voting)를 디코딩 레벨에서 적용하고 있다. 이것은 수신 노드에서 입력되는 신호의 최근 5개의 샘플링을 투표창(voting window) 내에 저장한 후, 저장되어 있는 1과 0의 개수에 따라서 입력값을 HIGH 또는 LOW로 판단하는 것으로, 보다 신뢰성 있는 전송을 가능하게 한다.

III. FlexRay 네트워크 설계 방법

FlexRay를 이용한 네트워크 설계는 두 단계의 절차를 거쳐 이루어 지게 되는데, 먼저 클러스터 설계 단계에서 통신 네트워크의 전반적인 구조를 결정하고, 다음으로 노드 설계 단계에서 각 노드의 세부적인 기능에 대해서 정의하는 절차로 이루어진다. 여기서, 클러스터 설계는 기본적인 네트워크 속도의 설정, 전반적인 통신 사이클 길이를 설정하는 단계이다. 특히, 마이크로틱의 길이(microtick length)와 마크로틱의 길이(macrotick length), 정적 구간과 동적 구간의 길이 등을 설정해야 한다.

노드 설계 단계에서는 네트워크를 구성하는 통신 노드들의 채널 설정과 슬롯 ID 할당, 그리고 동기화에 필요한 싱크 프레임의 ID 설정 등이 이루어 진다.

1. 클러스터 설계

클러스터 설계는 다수의 서브시스템으로 구성되어 있는 노드를 포함하여 전체적인 FlexRay 네트워크의 구조를 설계하는 과정이다. 이는 다음과 같은 단계를 거쳐서 이루어 진다.

가장 먼저, 네트워크에서 필요로 하는 시간 관련 파라미터를 설정한다. 여기서는 하드웨어로 구성되어 있는 노드에서 FlexRay 프로토콜의 실제적인 동작을 제어하는 통신 제어기의 오실레이터 클록(oscillator clock) 을 결정하고 버스 드라이버의 종류를 설정하는 과정을 포함한다. 다음으로 가장 중요한 설계 사양인 통신 속도와 통신 사이클 길이를 결정해야 하는데, 이는 각각의 노드에 전송되는 메시지의 성격과 시스템에서 요구되는 전송 지연 특성을 고려해서 결정되어야 한다.

두 번째 단계에서는 동기화와 관련된 설정을 수행해야 한다. 네트워크에서 사용되는 소구간의 길이와 오프셋을 정한 다음, 싱크 프레임의 수와 클록 동기화를 위한 보정 파라미터들의 값을 지정하는 과정이다. 이 때 FlexRay 클러스터 내에는 최소 2개, 최대 15개의 싱크 프레임이 존재할 수 있도록 설정해야 한다.

세 번째 단계에서는 정적 구간과 동적 구간을 설정하는 과정이다. 정적 구간을 구성하는 기본 시간 베이스로 정적 슬롯(static slot)의 길이와 오프셋을 지정한 다음, 정적 구간에서 전송될 데이터의 길이와 메시지 수를 결정하게 된다. 또한 이 과정에서 동적 구간의 사용 여부를 결정해 주어야 한다.

마지막으로 고려해야 할 요소로는 기본적인 통신 사이클의 구성과 관련되는 요소들을 정의하는 것이다. 사용하고자 하는 정적 구간, 동적 구간, 네트워크 유휴 시간, 심볼 윈도우들의 길이를 지정함으로써 전체적인 FlexRay 네트워크의 구조의 설계를 완료하게 된다. 그림 6에 이러한 과정을 거쳐 클러스터를 설계하는 절차를 나타내었다.

2. 노드 설계

노드 설계 단계는 클러스터 설계 단계를 거쳐 정의된 네트워크의 기본적인 구조를 바탕으로 각 노드별로 서브시스템을 설계하는 과정이다.

먼저, 네트워크를 구성하는 각 노드들의 사용 채널을 설정하고 노드별로 주요한 기능을 정의하게 된다. 특히 이 과정에서는 클러스터 켜기(wake-up) 역할을 하는 노드를 설정하여, 클러스터 켜기와 관련된 제어 메시지를 전송해야 할 채널을 선택하고 싱크 프레임으로 사용될 슬롯 ID를 지정해

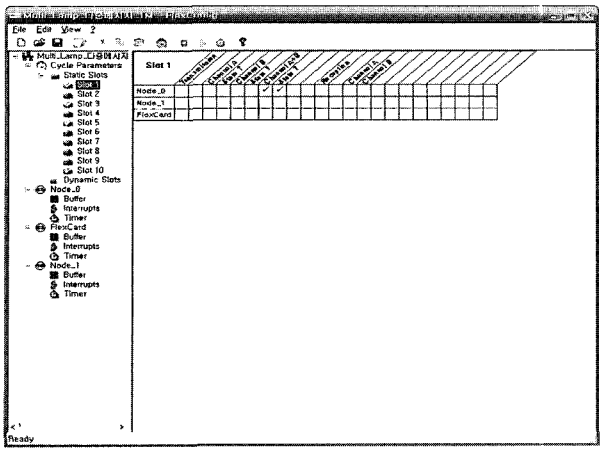


그림 7. 노드 설계 결과.
Fig. 7. Results of node design.

주어야 한다.

다음으로 클러스터 설계에서 이루어진 정적 구간과 동적 구간의 설정에 따라 각각 할당된 메시지 슬롯의 기능적인 측면을 정의해 주어야 한다. 이를 통해, 메시지의 송신 노드와 수신 노드 그리고 통신 채널의 설정이 이루어진다. 여기서 유의해야 할 사항은, 싱크 프레임으로 사용되는 메시지의 경우 반드시 두 개의 통신 채널로 동시에 전송되도록 해야 한다는 것이다. 그림 7에 이러한 과정을 거쳐서 완료된 노드 설계의 결과를 나타내었다.

클러스터 설계와 노드 설계를 거쳐 최종적으로 각각의 노드별로 CHI 파일을 생성하게 되면 비로소 FlexRay 네트워크를 사용할 수 있는 설계가 완료되게 된다. 이 CHI 파일은 클러스터 설계에 관련된 파라미터를 비롯하여 노드 설계 단계에서 이루어진 모든 구성 정보를 담고 있는 것으로 메인 프로그램과 함께 통신 제어기와 호스트에 다운로드함으로써 FlexRay 네트워크를 구현할 수 있게 된다.

IV. FlexRay 네트워크 구현 및 평가

1. FlexEntry를 이용한 FlexRay 네트워크 구현

본 절에서는 앞 장에서 서술한 설계 방법을 기반으로 하여 TZM 사의 FlexNode를 이용하여 FlexRay 네트워크를 구현하였다[7]. FlexNode는 호스트로 Freescale사의 16비트 마이크로 콘트롤러인 MC9S12DT256B를 사용하고 있으며, 트랜시버(transceiver)로 Philips사의 TJA1080을 사용하고 있다. 또한 통신 제어기로는 FlexRay Ver. 2.1을 지원하는 Bosch사의 E-Ray를 사용하고 있다.

FlexRay 네트워크 설계를 위하여, TZM 사의 FlexConfig를 사용하여 CHI 파일을 생성하였고, 메인 프로그램은 GNU 컴파일러를 이용하여, 그림 8과 같이 C언어로 작성하였다. 메인 프로그램에서는 FlexNode의 호스트와 통신 제어기의 연결을 위해 호스트의 포트 K를 이용하여 구현하였으며, 호스트와 버스 드라이버의 연결은 SPI 통신을 이용하였다. 이렇게 작성된 메인 프로그램은 콜드 스타트로 선정된 노드에서 싱크 프레임을 전송하여 동기화를 시도하고, 동기화가 완료되면 통신 사이클 내에서 자신에게 할당된 구간에서 데이터 프레임의 전송이 이루어지도록 하였다.

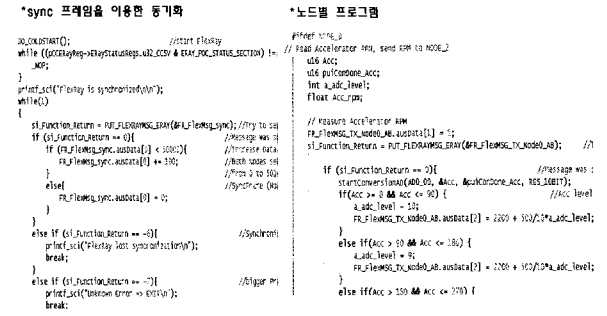


그림 8. FlexRay 구현을 위한 메인 프로그램.
Fig. 8. Main program of FlexRay implementation.

2. FlexRay 기반 테스트 베드 구현

구현된 FlexRay 네트워크 시스템의 성능을 평가하기 위해 모터 제어를 위한 테스트 베드를 구축하였다. 제어 대상이 되는 플랜트로는 DC 서보 모터가 사용되었으며 4개의 FlexRay 노드로 그림 9와 같이 구성하였다.

모터 제어를 위한 입력은 엑셀레이터(accelerator)와 브레이크(brake)가 달려있는 페달을 이용하여 FlexRay 네트워크에 접속되어 있는 2개의 노드를 통해 이루어진다. 또한, DC 서보 모터를 제어할 수 있는 제어 노드를 모터 노드와 별개로 두어 네트워크 지연 특성을 관찰할 수 있도록 하였다. 그리고 FlexRay의 채널 중복 특성을 관찰하기 위해서 모든 메시지는 두개의 채널을 사용하여 동시에 전송할 수 있도록 설계하였다.

엑셀레이터와 브레이크 페달을 누르면 누르는 정도에 따라 모터의 속도 목표가 각각 2200~2700RPM, 1700~2200RPM으로 설정 되도록 프로그램하였다. 입력되는 RPM값은 제어 노드로 전송되고 모터 노드에서는 DC서보 모터의 RPM을 측정하여 제어 노드로 전송한다. 제어 노드는 엑셀레이터 또는 브레이크에 의해 입력된 RPM값과 현재의 모터 RPM을 비교하여 기준 RPM을 추종할 수 있도록 PWM값을 계산하여 모터 노드로 전송한다. PWM 값을 받은 모터 노드는 이 값을 이용하여 DC서보 모터를 구동한다. 만약, 외부의 입력이 전혀 없는 초기 상태의 경우 모터는 2200RPM을 추종하도록 설계하였다.

FlexRay 네트워크의 데이터 전송 속도는 10Mbps, 통신 사이클은 5ms로 설정하였고, 그림 10과 같이 슬롯 ID 2, 3, 4를

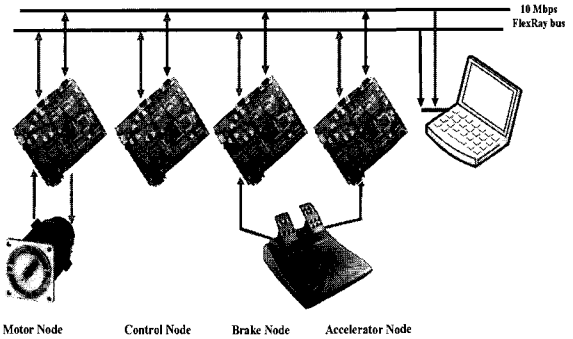


그림 9. FlexRay 기반 제어시스템 테스트 베드.
Fig. 9. Testbed of FlexRay based control system.

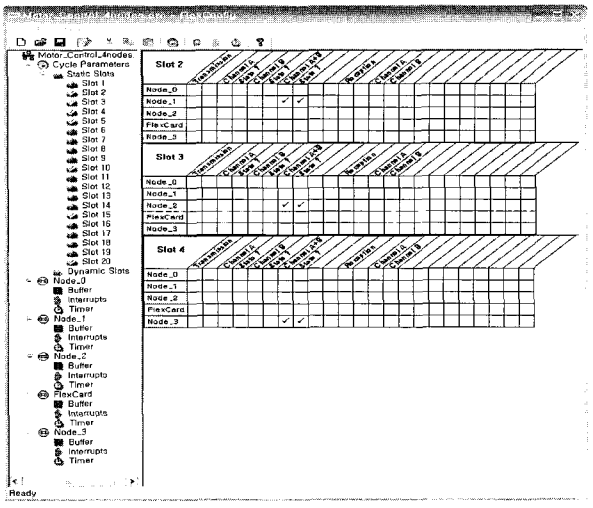


그림 10. FlexConfig를 이용한 싱크프레임의 슬롯 ID 할당.
Fig. 10. Assignment of syncframe slot ID using FlexConfig.

표 1. 슬롯 ID 할당.
Table 1. Assignment of slot ID.

Frame	Slot ID	Channel
Brake sync frame	2	A, B
Motor sync frame	3	A, B
Control sync frame	4	A, B
Accelerator RPM	5	A, B
Brake RPM	10	A, B
Motor RPM	15	A, B
PWM value	20	A, B

싱크 프레임으로 할당하였다. 그리고 모터 노드와 브레이크, 엑셀레이터의 RPM 값과 PWM 값의 전송을 위해서는 각각 5, 10, 15, 20의 슬롯 ID를 부여하였고 표 1에서 FlexConfig를 통한 슬롯 ID 할당의 결과를 나타내었다. 이렇게 설정된 결과로 그림 11과 같은 통신 사이클을 구성하여 제어 시스템에 적용하였다[9].

3. 테스트 베드를 이용한 FlexRay 시스템 평가

FlexRay 기반 제어 시스템의 성능을 확인하기 위해서 Vector 사의 CANoe.FlexRay를 이용하였다[8]. 먼저 FlexRay 버스 상에 전송되는 데이터의 시그널 접근을 통한 모니터링을 하기 위해서 FIBEX(Field Bus Exchange Format) 파일을 통한 가상 노드를 생성하였다. 왜냐하면, 현재까지는 FlexRay의 버스 모니터링을 위해서는 CAN database를 변환해서 사용해야만 하는데 이를 위해서는 서로 다른 프로토콜의 환경변수를 조정하기 위한 추가적인 작업이 필요하기 때문이다. 따라서 FlexRay의 네트워크와 관련된 여러 가지 파라미터들과 실제 구현된 제어 시스템을 위해 사용된 여러가지 메시지들에 대한 변수들을 새롭게 정의하기 위해 그림 12와 같은 FIBEX 파일을 생성하였다.

다음으로 제어 입력인 브레이크와 엑셀레이터의 기준 RPM, 그리고 제어 출력인 DC모터의 출력 RPM의 변화 양상을 모니터링 하기 위해서 CANoe.FlexRay의 Panel 및 Graphics 기능을 이용하였다. 실제 자동차의 계기판을 모사하였고CAN

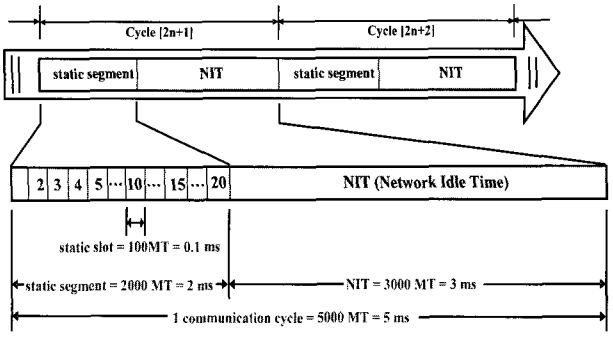


그림 11. 완성된 통신 사이클.
Fig. 11. Completed communication cycle.

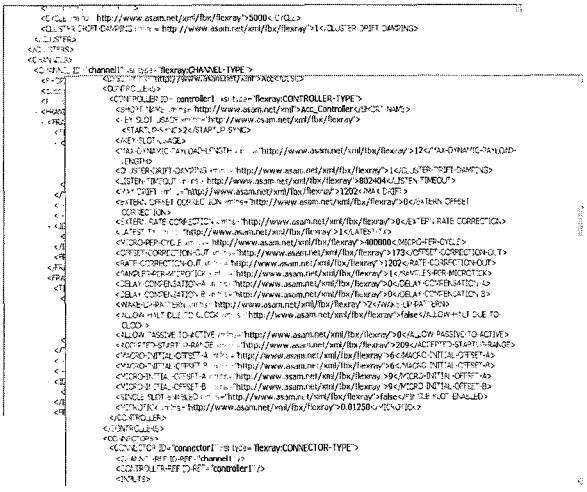


그림 12. FlexRay 모니터링을 위해 생성한 FIBEX.
Fig. 12. Generated FIBEX for FlexRay monitoring.

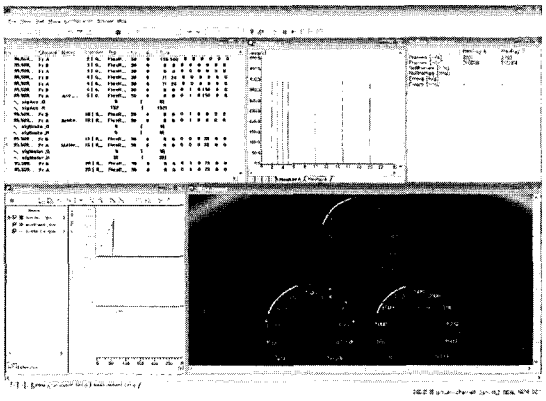


그림 13. 시스템 평가를 위해 구현한 모니터링 시스템.
Fig. 13. Monitoring system for system evaluation.

database에서 정의한 환경 변수를 삽입하여 그림 13과 같이 브레이크와 엑셀레이터의 입력에 의해 DC모터의 출력 RPM의 변화를 모니터링 할 수 있도록 구현하였다.

이를 이용하여 실제적인 FlexRay 네트워크 시스템의 성능은 그림 14와 같이 브레이크와 엑셀레이터의 입력에 따른 모터의 RPM 추종 성능을 평가하는 형태로 이루어졌다. 먼저 그림 15는 엑셀레이터 페달을 총 6회 밟았을 때, 모터의 RPM 추종 양상을 보여주고 있다. ①과 같이 페달을 힘껏 밟

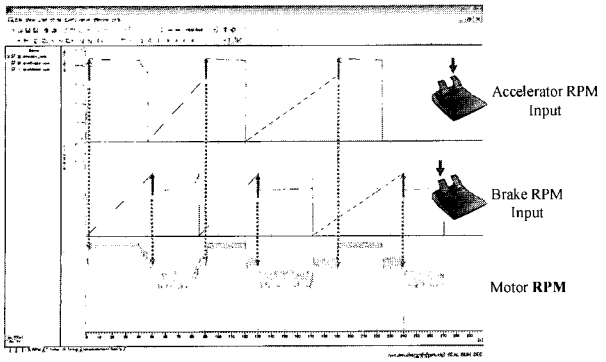


그림 14. 기준 입력에 따른 모터 RPM 추종 성능 결과.
Fig. 14. Result of control performance depend on reference input.

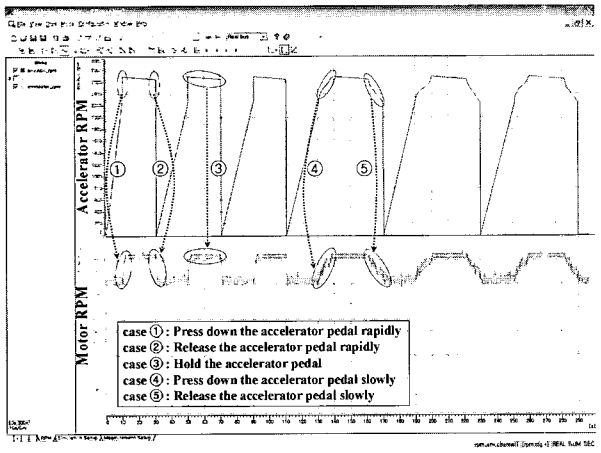


그림 15. 엑셀레이터 입력에 따른 DC 모터 출력 RPM.
Fig. 15. Output RPM of DC motor on acceleration input.

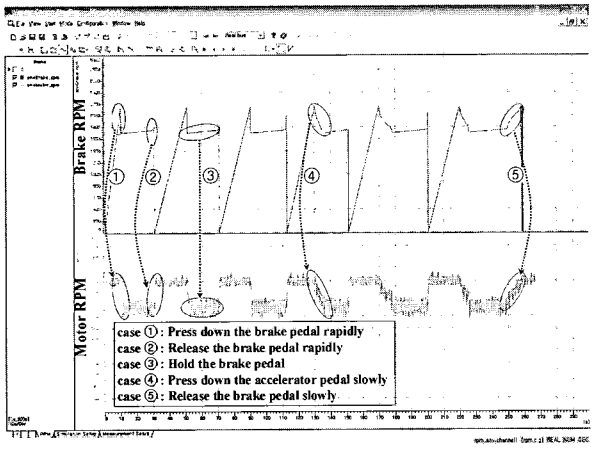


그림 16. 브레이크 입력에 따른 DC 모터 출력 RPM.
Fig. 16. Output RPM of DC motor on braking input.

아서 20초 동안 이 상태를 유지하면, 엑셀레이터 노드에서는 최대 기준 입력이 제어 노드로 전송되게 된다. 이 값을 제어 노드로부터 받은 모터는 최대 속도인 2700RPM으로 가속하게 된다. 그 후, 밟고 있던 엑셀레이터 패달에서 재빨리 발을 떼어 입력이 없는 초기 조건으로 돌아가도록 하였다(②). 그 결과, 모터는 약간의 오차 범위 내에서 초기 설정 속력인 2200RPM으로 회전하였다. 다음으로 ④와 같이 엑셀레이터 페달을 서서히 밟아 엑셀레이터 노드를 통한 기준 입력값을 서서히 증가시킬 때, 제어 노드에 의해 모터 또한 2200RPM에서 2700RPM으로 서서히 속도가 증가하는 것을 확인할 수 있었다.

그림 16은 동일하게 브레이크 페달을 통한 입력에 따른 모터의 RPM추종 양상을 보여주고 있다. 먼저, 브레이크 페달을 힘껏 밟아서 20초 동안 이 상태를 유지하여 브레이크 노드의 최대 입력이 제어 노드로 전송되도록 하였다. 이 값을 제어 노드로부터 받은 모터는 역시 1700RPM의 속력으로 감속됨을 확인할 수 있었다. 또한 ④, ⑤의 조건처럼 브레이크 페달을 서서히 밟아 입력을 서서히 주거나, 페달 입력을 제거하여 초기 조건으로 돌아가게 했을 경우, 역시 모터는 브레이크 노드에 의해 입력되는 값을 기초로 하여 제어 노드를 통해 기준 입력값에 근접하게 추종하는 것을 확인할 수 있었다.

V. 결론 및 향후 계획

본 논문에서는 네트워크 기반 실시간 분산 제어 시스템을 위하여 시간 기반 프로토콜의 한 종류인 FlexRay 네트워크 설계 방법을 제시하였다. 또한, FlexRay 네트워크 설계 방법을 이용하여 모터 제어 시스템을 구현하였다. 그리고, FIBEX 파일을 생성하여 CANoe.FlexRay를 통한 모니터링이 가능하도록 하여 브레이크나 엑셀레이터 입력에 의해 모터 시스템이 기준 입력에 추종하여 동작됨을 확인하였다.

분산형 제어 시스템의 구현 과정은 매우 많은 시간이 요구되는 작업이지만, FlexRay를 이용한 시스템의 설계는 보다 쉽게 구현이 가능하며, 또한 다수의 제어시스템이 사용되는 응용에서도 효과적으로 적용이 가능함을 확인하였다.

그러나 본 논문에서는 4개의 FlexRay 노드만을 이용하여 제어 시스템을 구성하였으며, 한 개의 모터를 이용한 시스템이었다. 따라서, 향후 지능형 자동차와 같이 보다 많은 통신 노드와 제어 시스템을 필요로 하는 분야에서의 적용을 위해서는 다수의 ECU 들로 구성된 시스템을 통한 평가와 효과적인 제어 기법에 대한 연구가 필요할 것이다. 또한, 안전성 및 신뢰성과 직결되는 정시성이란 측면에서 FlexRay 네트워크가 가지는 통신 지연 뿐만 아니라 지터와 관련한 연구도 병행되어야 할 것이다.

참고문헌

[1] G. Leen and D. Hefferman, "Digital networks in the automotive vehicle," *IEEE Computer and Control Engineering Journal*. vol. 10, no. 6, pp. 257-266, Dec. 1999.

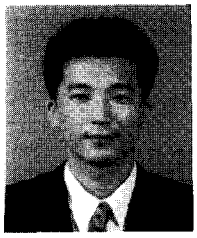
[2] T. F. Abdelzaher and K. G. Shin, "Combined task and message scheduling in distributed real-time systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 11, pp. 1179-1191, Nov.1999.

[3] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, "Trends in automotive communication systems," *Proceedings of the IEEE* vol. 93, no. 6. pp. 1204-1223, Jun. 2005.

[4] C. Quigley, "Electronic system integration for hybrid and electric vehicles," *Hybrid Vehicle Conference, IET The Institution of Engineering and Technology*, pp. 125-140, 2006.

[5] FlexRay consortium. FlexRay protocol specification 2.1 revision

A, 2005.
[7] FlexEntry, FlexConfig, http://www.TZM.com.
[8] CANoe.FlexRay, http://www.vector-informatick.com
[9] I. Park, M. Shin, J. Youn, J. Ma, M. Sunwoo, "A study on a design of distributed control systems based on a FlexRay network," *2006 Fall conference proceeding* of KSAE, pp. 1287-1293, 2006.

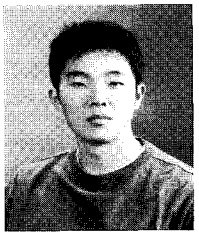


하 경 남

1974년 9월 10일생. 2001년 부산대학교 기계공학부 졸업. 2001년~2003년 BOSCH 근무. 2007년 부산대학교 박사과정 수료. 2007년 현재 기계공학연구정보센터 전임연구원. 관심분야는 산업용 네트워크, 차량용 네트워크.

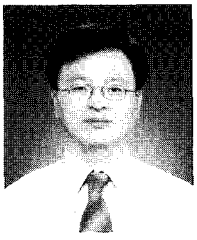
이 석

1961년 12월 11일생. 1984년 서울대학교 기계공학과 졸업. 1985년 펜실바니아 주립대학교 석사. 1990년 동 대학원 박사. 1990년~1993년 신시내티 대학교 기계공학과 조교수. 1993년~현재 부산대학교 기계공학부 교수. 관심분야는 산업용 네트워크, 차량용 네트워크, 홈 네트워크, 센서 네트워크.



이 원 석

1979년 5월 16일생. 2006년 부산대학교 기계공학부 졸업. 2006년~현재 부산대학교 지능기계공학과 석사과정 재학중. 관심분야는 차량용 네트워크, 분산제어 시스템.



이 경 창

1971년 5월 1일생. 1996년 부산대학교 생산기계공학과 졸업. 1998년 동 대학원 석사. 2003년 동 대학원 박사. 1998년~2003년 기계공학연구정보센터 전임연구원. 2003년~2005년 울산대학교 네트워크기반 자동화 연구센터 전임연구원. 2005년~현재 부경대학교 전기제어공학부 조교수. 관심분야는 산업용 네트워크, 차량용 네트워크, 홈 네트워크, 센서 네트워크.