매틀랩/시뮬링크 기반 플렉스레이 네트워크 시스템의 구현 및 검증

Implementation and Verification of FlexRay Network System using Matlab/Simulink

윤 승 현, 서 석 현, 황 성 호, 권 기 호, 전 재 욱*
(Seung-Hyun Yoon, Suk-Hyun Seo, Sung-Ho Hwang, Key Ho Kwon, and Jae Wook Jeon)

Abstract: As increasing the number of Electronic Control Units in a vehicle, the proportion for reliability and stability of the software is going increasingly. Accordingly, the traditional CAN network has occurred the situation that the requirement of developing vehicle software is not sufficient. To solve these problems, the FlexRay network which is ensured the high bandwidth and real-time is generated. However it is difficult to implement FlexRay based application software because of complex protocol than traditional CAN network. Accordingly the system for analysis and verification of network state is needed. Also vehicle vendor develops application software using Matlab/Simulink in order to increase productivity. But this development method is hard to solve the network problem of node to node. Therefore this paper implements Matlab/Simulink based FlexRay network system and verifies it through comparing with existing embedded system.

Keywords: matlab, simulink, FlexRay, simulation, verification

I. 서론

1990년대 중반의 한 대의 자동차에 내장된 전자 제어 유닛의 수가 10개에 불과 했던 것이 2000년대 들어오면서 50개 이상으로 늘어났다. 더불어 미래형 자동차에는 지금보다 훨씬 많은 수의 전자 제어 유닛이 장착될 예정이다. 이렇게 차량 내 전자 제어시스템의 비중이 증가하고 복잡해짐에 따라 안정성과 신뢰성의 요구가 증가되고 있다. 이러한 요구를 만족하기 위해서는 높은 대역폭을 가지면서 오류에 강인한 통신 프로토콜이 필요하게 되었다. 기존에 차량용 네트워크로 주로 사용되고 있는 CAN (Controller Area Network)은 속도가 낮으며 응답속도가 비확정적이기 때문에 샤시 제어와 같은 실시간성을 요구하는 네트워크로 사용하기에는 적합하지 않았다. 이러한 문제를 해결하기 위해 차세대 차량 통신 프로토콜인 플렉스레이가 개발되어 여러 자동차 업계를 중심으로 점차 확대 적용되고 있다. 하지만 플렉스레이는 CAN 네트워크에 비해 프로토콜이 복잡하기 때문에 네트워크를 기반으로 하는 응용 소프트웨어를 구현하기가 어려운 문제점을 가지고 있다. 이러한 문제점을 해결하기 위해서는 기존에 만들어진 전자 제어 유닛 소프트웨어와 쉽게 연동하면서 네트워크의 상태의 분석하고 수정에 따른 결과를 쉽게 확인할 수 있는 환경을 제공할 수 있어야 한다[1-3].

전자 제어 시스템에 들어가는 소프트웨어가 점차 복잡해짐에 따라 개발 및 유지 보수에 소요되는 시간과 비용이 크게 증가하게 되면서 자동차 업체에서는 모델 기반을 통한 빠른 프로토타이핑 개발 기법을 사용하기 시작했다. 이는 전자 제어 알고리즘의 개발 뒤에 따르는 검증 및 구현 작업을 모델 기반 자동 코드 생성기술을 이용하여 자동으로 수행하는 것을 의미한다. 이러한 기술은 개발된 알고리즘을 수정시에 자동 코드 생성기에 의해 프로그램 코드가 생성되므로 시간과 비용을 절약할 수 있게 한다. 따라서 이러한 모델 기반 개발을 위해 매틀랩/시뮬링크 개발 환경이 주로 사용되고 있다. 하지만 매틀랩/시뮬링크 개발 환경에서는 각 전자 제어 시스템에서 외부 네트워크로 메시지를 송/수신 하기 전까지의 과정에 대한 자동 코드 생성 기능 만을 지원함에 따라 실제로 분산 네트워크 시스템 개발시 노드 간에 발생할 수 있는 네트워크 문제를 해결하기 어려운 단점을 가지고 있다[4].

따라서 본 논문에서는 매틀랩/시뮬링크를 기반으로 하는 플렉스레이 네트워크 시스템을 구현하고 이러한 시스템을 기반으로 다양한 프로토콜 설정에 따른 플렉스레이 네트워크를 시뮬레이션하며 이를 실제 임베디드 시스템에 적용해 검증하고자 한다.

II. 관련 연구

플렉스레이는 BMW와 Daimler Chrysler, Freescale, Philips 등이 참여하는 컨소시엄에 의하여 제정된 분산 실시간 차량용 네트워크 프로토콜이다. 플렉스레이는 최대 10Mbps의 전송속도를 지원하며 두 개의 채널을 가지고 있어 이중화를 지원한다. 또한 주기적으로 전송하는 static 메시지와 비주기적으로 전송하는 dynamic 메시지를 모두 지원하며 버스나 스타 및 하이브리드 토폴로지의 사용을 보장하는 유연성을 가지고 있다. 더불어 전송 지연이 확정적이고 예측 가능하다는 특징을 가지고 있다[5].

플렉스레이의 메시지 프레임은 그림 1과 같이 크게 header segment, payload segment 그리고 trailer segment로 나뉘어 있다. header segment는 5바이트로 구성되어 있는데 프레임의 상태를 나타내는 4개의 indicator 비트(payload preamble, null frame,

* 책임저자(Corresponding Author)
논문접수: 2010. 3. 15., 수정: 2010. 4. 15., 채택확정: 2010. 4. 30.
윤승현, 서석현, 권기호, 전재욱: 성균관대학교 전자전기컴퓨터공학부
(ibmlov@gmail.com/imakeit@ece.skku.ac.kr/khkwon@ece.skku.ac.kr/jwjeon@yurim.skku.ac.kr)
황성호: 성균관대학교 기계공학부(hsh@me.skku.ac.kr)
※ 본 과제(연구)는 지식경제부와 한국산업기술진흥원의 전략기술인력양성사업으로 수행된 결과임.

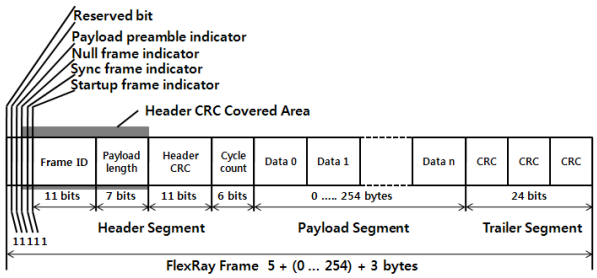

그림 1. 플렉스레이 메시지 프레임 포멧.

Fig. 1. FlexRay message frame format.

sync frame, startup frame)와 11비트의 프레임 ID, 7비트의 Payload length, 11비트의 header CRC 그리고 6비트의 Cycle count 값을 포함하고 있다. Payload segment는 0에서부터 254바이트까지의 데이터를 포함할 수 있다. Trailer segment는 Header segment와 Payload segment의 모든 데이터 값에 대한 24비트의 CRC 값을 포함한다[6].

플렉스레이 네트워크에서 통신 사이클은 주기적인 메시지 전송이 가능한 static 구간과 비주기적인 메시지 전송이 가능한 dynamic 구간, 프로토콜에 정의된 symbol을 전송하는 symbol window 그리고 네트워크 시간 동기화를 수행하는 network idle time으로 이루어져 있다[6].

그림 2는 static 구간에서의 메시지 전송 제어를 나타내고 있다. Static 구간에서는 time-triggered 방식을 사용해서 메시지를 전송한다. static 구간은 일정한 길이를 가지고 있는 static slot으로 나뉘어 있으면 각 노드에서의 static 메시지의 id 값에 따라 해당하는 static slot 구간에서 실제 전송이 이루어진다. 그림 3은 dynamic 구간에서의 메시지 전송 제어를 나타내고 있다. dynamic 구간에서는 event-triggered 방식을 사용해서 메시지를 전송한다. 이 구간에서는 작은 길이를 가지고 있는 mini slot에 따라 메시지를 전송 한다. static 구간과 같이 전송하는 dynamic 메시지의 id 값과 일치하는 slot counter를 갖는 mini slot 구간에서 실제 전송이 이루어지지만 static 구간과는 달리 dynamic 메시지 프레임의 크기에 따라 하나 이상의 mini slot을 차지할 수 있으며 slot counter 또한 dynamic 메시지를 전송하고 있는 중에는 증가하지 않고 전송이 완료되고 나서야 다시 증가한다. 예를 들어 id 값이 5와 7을 가지고 있는 dynamic 메시지들이 전송될 때 먼저 mini slot counter가 5가 되면 첫 번째 메시지가 전송이 되고 이 메시지 전송이 완료되고 나서 다시 mini slot counter가 2가 증가해야 다음 메시지가 전송될 수 있다[6].

플렉스레이 네트워크의 각 노드들은 인코딩/디코딩을 위해

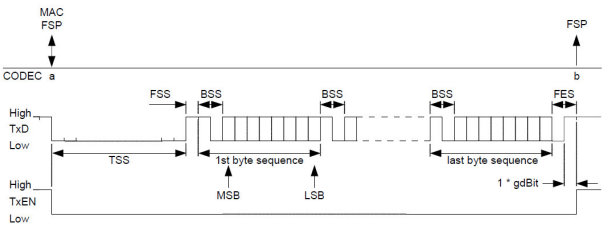

그림 2. Static 구간에서의 메시지 전송 제어.

Fig. 2. Message transmission control in static segment.

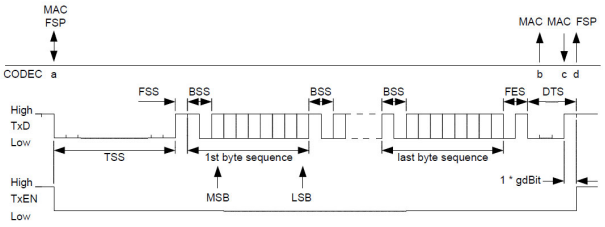

그림 3. Dynamic 구간에서의 메시지 전송 제어.

Fig. 3. Message transmission control in dynamic segment.

non-return to zero(NRZ) 신호 방식을 사용하며 물리적 매체에 존재하는 비트 스트림으로부터 프레임과 심볼 정보를 생성 또는 추출 해내고 이러한 정보를 관련된 다른 프로세스에 전달한다. 먼저 인코딩은 전송 구간 (static 구간, dynamic 구간)에 따라서 과정이 달라지는데 공통적으로는 Transmission start sequence, Frame start sequence, Byte start sequence, Frame end sequence를 거치면서 인코딩 과정이 진행된다. 먼저 Transmission start sequence에서는 네트워크를 통한 연결 설정을 초기화하는데 사용되며 파라미터 값에 따라 연속적인 LOW 값들(TSS 비트)을 생성해서 전송 프레임 앞에 추가한다. Frame start sequence에서는 TSS 비트 값들과 전송 프레임 사이에 한 비트의 HIGH 값(FSS 비트)을 추가한다. Byte start sequence에서는 수신 받는 디바이스 쪽에서의 비트 스트림 타이밍 정보를 제공하기 위해 전송 프레임의 1바이트마다 앞에 연속적인 HIGH 1비트와 LOW 1비트 값(BSS 비트)을 추가한다. Frame end sequence에서는 전송 프레임의 마지막 비트 뒤에 연속적인 Low 1비트 하나와 HIgh 1비트(FES 비트)를 추가한다. 여기에 dynamic segment인 경우에는 dynamic trailing sequence 과정이 추가되어 FES 비트 다음에 연속적인 LOW 3비트와 HIGH 1비트 값(DTS 비트)을 추가한다[6].

III. 시스템 구현

구현된 플렉스레이 네트워크 시스템은 크게 노드 별로 메시지를 전송하는 transfer subsystem과 메시지를 수신하는 receiver subsystem, 플렉스레이 프로토콜에 따른 각 구간 별 전송 제어를 담당하는 media access control subsystem, 노드들을 네트워크로 연결하는 bus subsystem으로 구성되어 있다. 각 시스템에 대한 설명은 다음과 같다.

1. Transfer subsystem

Transfer subsystem은 크게 header segment를 생성하는

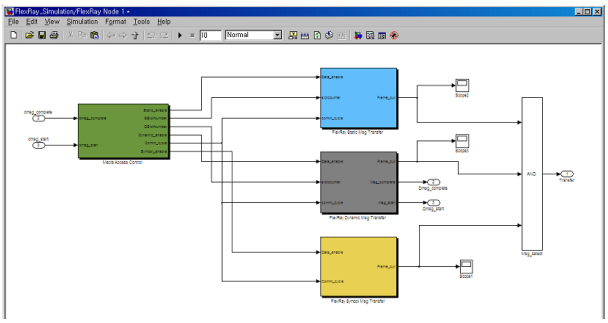

그림 4. Transfer subsystem.

Fig. 4. Transfer subsystem.

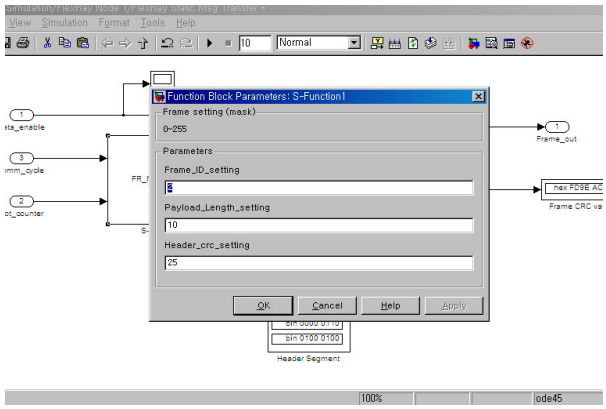

그림 5. Header generator 블록에서 사용자 파라미터 입력.
Fig. 5. User parameter input in header generator block.

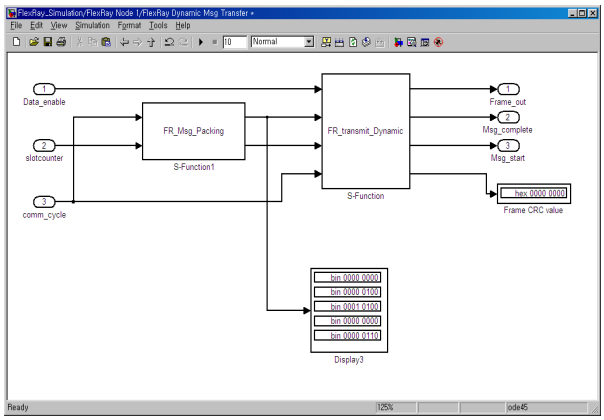

그림 6. Dynamic transfer 블록.
Fig. 6. Dynamic transfer block.

message packing 블록, payload segment를 생성하는 payload generator 블록, static 메시지를 전송하는 static transfer block, dynamic 메시지를 전송하는 dynamic transfer 블록, symbol 메시지를 전송하는 symbol transfer 블록으로 나뉘어져 있다. 그림 4는 transfer subsystem을 나타내고 있다.

1.1 Header generator 블록

사용자 파라미터 입력을 받아 전송할 플렉스레이 메시지의 header segment를 생성한다. 입력 받는 파라미터 값은 frame ID, payload length, header CRC로서 frame ID는 전송 프레임의 ID 값이고 payload length는 전송 데이터의 크기, header CRC는 header segment의 CRC 값을 나타낸다. 이러한 파라미터 값은 그림 5와 같이 해당 블록을 더블 클릭해서 생성되는 창을 통해 입력 받는다.

1.2 Payload generator 블록

사용자 파라미터 입력을 받거나 외부 입력 블록에서 값을 받아 payload segment를 생성한다.

1.3 Static transfer 블록

Static transfer 블록은 플렉스레이 네트워크에서 static segment 구간에 전송되는 메시지를 제어한다. Header generator 블록으로부터 받은 header segment와 payload generator 블록으로부터 받은 payload segment, 그리고 이 두 segment의 CRC 값이 저장된 trailer segment를 조합해서 플렉스레이 메시지 프레임을 생성한다. 생성된 메시지 프레임은 static transfer 블록이 한번 실행될 때마다 1 바이트씩 bus subsystem으로 전송된다. 단 static 메시지가 전송되는 구간에서만 전송하기 위해서 추가적으로 두 신호 값(data_enable, slot_enable)을 입력 받아 메시지 전송 유무를 제어한다. Slot_enable 신호를 통해 현재의 static slot 에서의 전송유무가 결정되면 data_enable 신호를 통해 해당 슬롯 구간에서 메시지가 전송되는 클럭 타이밍이 결정된다.

1.4 Dynamic transfer 블록

Dynamic transfer 블록은 플렉스레이 네트워크에서 dynamic segment 구간에 전송되는 메시지를 제어한다. 그림 6은 이러한 dynamic transfer 블록을 나타내고 있다. 기본적으로 static transfer 블록과 비슷하게 구현되어 있으며 time-triggered 방식인 static transfer 블록과는 달리 event-triggered 방식으로 전송하기 위해 두 개의 신호 (msg_start, msg_complete)가 추가 되었다. 각각의 신호 값은 버스 상에 다른 dynamic 메시지 전송의 시작과 끝을 알리는 역할을 한다. 이 두 신호 값을 통해 dynamic transfer 블록에서 메시지 전송 타이밍을 결정한다.

또한 일정한 주기 간격으로 메시지를 전송하는 static transfer 블록과는 달리 비주기적으로 전송하는 dynamic 메시지를 구현하기 위해 전송되는 cycle을 파라미터 입력 값으로 설정할 수 있도록 구현했다

1.5 Symbol transfer 블록

Symbol transfer 블록은 플렉스레이 네트워크에서 symbol window 구간에 전송되는 메시지를 제어한다. Dynamic transfer 블록과 같이 전송되는 cycle을 파라미터 입력 값을 설정하며 wakeup 메시지나 CAS/MTS 메시지 중 하나를 선택해서 전송하도록 구현했다.

2. Receiver subsystem

Receiver subsystem은 각 노드 별로 수신 동작을 구현하며 수신 받은 메시지 프레임을 디코딩 하는 message_decoding 블록과 디코딩 된 메시지를 분석해서 static 메시지와 dynamic 메시지로 구분해서 출력하는 message_unpacking 블록으로 이루어져 있다. 그림 7은 이러한 receiver subsystem을 나타내고 있다.

2.1 Message decoding 블록

Message decoding 블록은 실행될 때마다 버스로부터 1바이

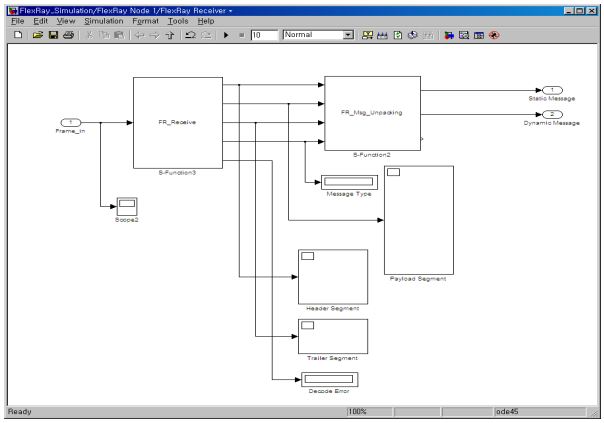

그림 7. Receiver subsystem.
Fig. 7. Receiver subsystem.

트씩 문자를 수신 받는다. 수신 받은 문자는 플렉스레이 프로토콜에 따라 디코딩 과정을 수행하고 디코딩 과정 중에 발생하는 에러를 감지하여 관련 정보를 출력할 수 있도록 구현하였다. 하나의 메시지 프레임에 대한 디코딩 과정이 완료될 때마다 header segment, payload segment, trailer segment로 나눠서 message unpacking 블록으로 전달한다.

2.2 Message unpacking 블록

Message unpacking 블록은 수신 받은 메시지의 종류(static, dynamic, symbol)를 판단하고 message decoding 블록으로부터 받은 header segment 데이터를 분석하여 payload segment의 길이를 판단한 뒤에 메시지 출력 버퍼에 payload segment 값을 저장한다. 이 버퍼 값은 추가로 연결된 Output 블록(LED, 7-segment, LCD 등)의 입력 값으로 사용하게 된다.

3. Media Access Control subsystem

Media Access Control subsystem은 플렉스레이 통신 사이클을 이루고 있는 static 구간, dynamic 구간, symbol window 그리고 network idle time 구간 동안 전송되는 메시지를 관리하는 역할을 수행한다. 이 subsystem은 노드 별로 존재하며 각각의 노드 간에 설정된 메시지들의 종류(static, dynamic, symbol)에 따라 해당하는 전송 타이밍이 되면 각 메시지 전송 블록(static transfer, dynamic transfer, symbol transfer)에게 신호 값을 보낸다. 먼저 static_enable 신호는 static segment 구간이 시작되었다는 것을 알린다. SSlotNumber는 현재 전송 가능한 static slot number를 알린다. Dynamic_enable 신호는 dynamic segment 구간이 시작되었다는 것을 알리고 DSlotNumber는 현재 전송 가능한 dynamic mini-slot Number를 알린다. Symbol_enable 신호는 symbol_window 구간이 시작되는 것을 알리고 comm_cycle 신호는 현재의 통신 cycle 값을 전달한다.

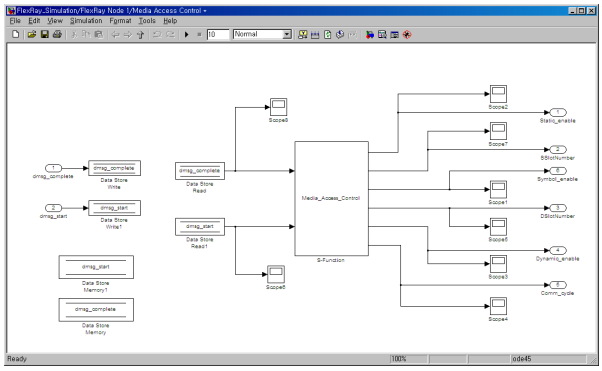

그림 8. Media access control subsystem.

Fig. 8. Media access control subsystem.

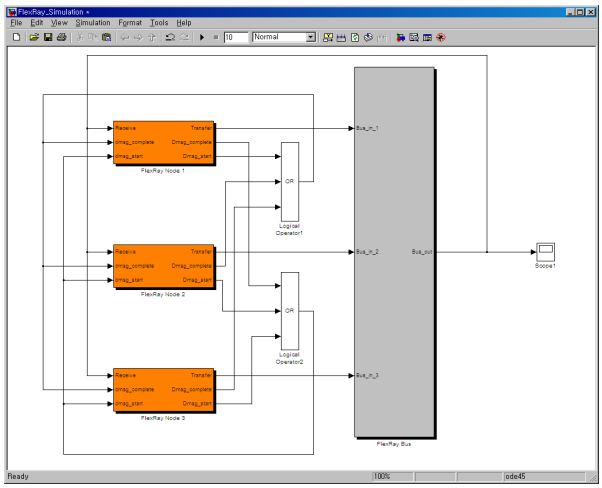

그림 9. 플렉스레이 네트워크 시스템.

Fig. 9. FlexRay network system.

4. Bus subsystem

연결된 여러 노드 사이에서 메시지 데이터 교환이 가능하도록 구현하였다. 기본적으로 노드 개수만큼 입력 값을 받으며 이 입력 값은 bus subsystem이 실행될 때마다 1바이트씩 갱신된다. 입력 값들은 idle 시에는 HIGH 값을 가지고 있기 때문에 노드 별로 입력 받는 값들을 AND 연산을 통해 multiplexing이 가능하도록 구현한다.

IV. 실험 및 결과

그림 9는 구현된 플렉스레이 네트워크 시스템의 전체적인 모습을 나타내고 있다. 회색의 블록은 네트워크 버스를 나타내며 주황색의 블록은 각각 노드를 의미한다. 노드 블록은 네트워크 구성에 따라 추가로 생성이 가능하도록 되어 있다. 여기서는 세 개의 노드를 가지고 표 1, 2, 3과 같은 설정 하에

표 1. 플렉스레이 네트워크 시스템 설정.

Table 1. FlexRay network system configuration.

	설정 값
Static Slot 수	20
Static Slot 간격 (macrotick)	200
Mini slot 수	20
Mini Slot 간격 (macrotick)	30
Symbol Window 간격 (macrotick)	100
Network Idle Time 간격 (macrotick)	200

표 2. 플렉스레이 네트워크 Static 구간 메시지.

Table 2. FlexRay network static segment message.

	노드 1	노드 2	노드 3
ID	1	4	12
Payload Length	16	16	16
DATA0	0x34	0x78	0xBC

표 3. 플렉스레이 네트워크 Dynamic 구간 메시지.

Table 3. FlexRay network Dynamic segment message.

	노드 1	노드 2	노드 3
ID	64	63	45
Payload Length	8	8	8
DATA0	0x22	0x33	0x44

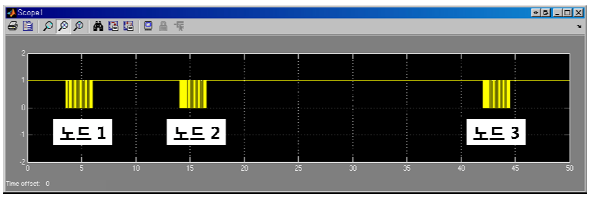

그림 10. static 메시지 전송 시뮬레이션 결과.

Fig. 10. static message transmission simulation result.

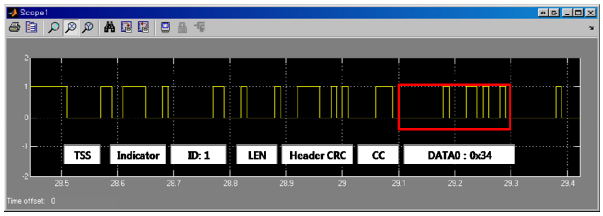

그림 11. 노드 1의 static 전송 메시지.
Fig. 11. Node 1's static transmission message.

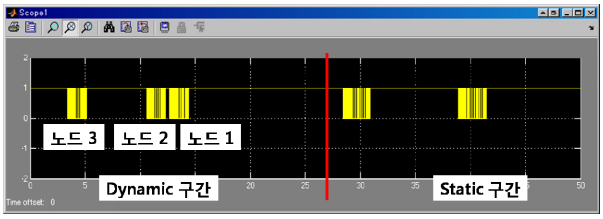

그림 12. Dynamic 메시지 전송 시뮬레이션 결과.
Fig. 12. Dynamic message transmission simulation result.

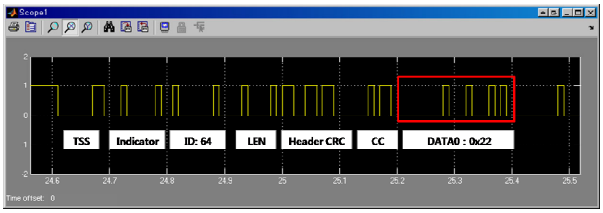

그림 13. 노드 1의 dynamic 전송 메시지.
Fig. 13. Node 1's dynamic transmission message.

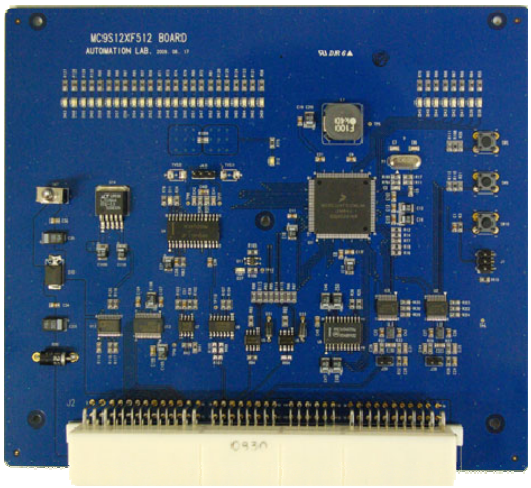

그림 14. MC9S12XF 임베디드 시스템.
Fig. 14. MC9S12XF Embedded System

시뮬레이션을 진행하였다.

그림 10은 노드 1~3에서 발생한 센서 값이 static 구간에서 주기적으로 전송되는 시뮬레이션 결과를 나타낸 것이다. 노드 1의 static 메시지를 자세히 확대해보면 그림 11에서와 같이 설정된 값에 따라 메시지 전송이 진행된 것을 확인할 수 있다. 그림 12는 노드 1~3에서 발생한 이벤트 값이 dynamic 구간에서 주기적으로 전송되는 시뮬레이션 결과를 나타낸 것이다. 노드 1의 메시지를 자세히 확대해보면 그림 13에서

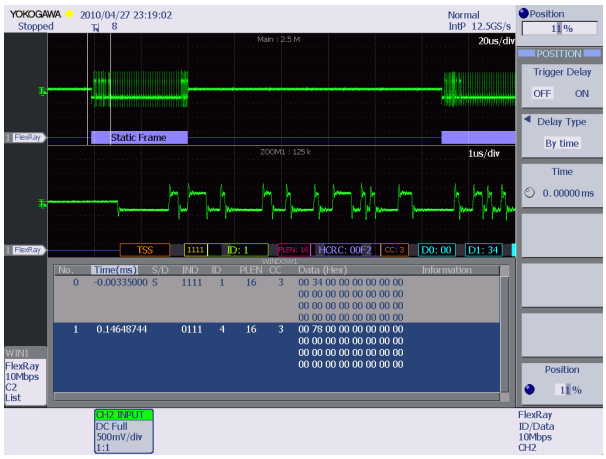

그림 15. 버스 분석기를 통한 static 전송 메시지의 확인.
Fig. 15. Checking static message using bus analyzer.

그림 16. 버스 분석기를 통한 dynamic 전송 메시지의 확인
Fig. 16. Checking dynamic message using bus analyzer

와 같이 설정된 값에 따라 메시지 전송이 진행된 것을 확인할 수 있다.

시뮬레이션 결과를 검증하기 위해 그림 14와 같은 Freescale 사의 16비트 MC9S12XF 마이크로 컨트롤러를 사용한 플렉스레이 기반 임베디드 시스템을 가지고 시리얼 버스 분석기를 통해 버스 상의 신호를 측정하였다. 시뮬레이션과 동일하게 세 개의 MC9S12XF 보드를 연결해 플렉스레이 네트워크를 구성하였으며 버스 상의 세가지 신호(BM, BP, GND)를 시리얼 버스 분석기에 연결해서 발생되는 파형을 측정하였다.

그림 15는 시리얼 버스 분석기를 통해 static 구간의 신호를 측정한 것을 스크린 캡쳐 한 것이다. 노드 1 메시지(ID: 1)를 확대한 아래 파형을 보면 그림 11의 시뮬레이션 결과와 동일한 것을 확인할 수 있다. 그림 16에서 위 파형은 dynamic 구간에서 전체 메시지 전송을 나타내고 아래 파형은 노드 1의 dynamic 메시지(ID: 64) 전송을 확대한 파형을 나타내고 있다. 시뮬레이션과 동일하게 dynamic 구간에서 ID 값에 따른 중재로 노드간 메시지 전송 순서가 결정되어 노드 2의 dynamic 메시지(ID: 63)가 먼저 전송된 것을 확인할 수 있다.

V. 결론

본 논문에서는 플렉스레이 네트워크 시스템을 구현 및 시뮬레이션하고 이를 검증하고자 했다. 매틀랩/시뮬링크를 이용해서 static 메시지와 dynamic 메시지를 생성하는 부분과 이러한 생성된 메시지의 종류에 따라 해당하는 구간에서 송수신이 발생하도록 제어하는 부분, 그리고 실제 송수신이 이루어지는 부분을 모델링 하여 네트워크 시스템을 구현하고 프로토콜 관련 파라미터 설정 값을 입력 받아 해당하는 결과를 시뮬레이션 할 수 있도록 하였다. 이러한 시뮬레이션 결과는 임베디드 시스템을 통해 실제 신호를 측정하여 비교함으로써 검증하였다. 구현된 모델 기반 플렉스레이 네트워크 시스템은 차량 제어 로직과 연동함으로써 실제 차량용 네트워크 시스템을 구축하기 전에 다양한 시뮬레이션을 통해 개발 시간 단축 및 비용 절감에 도움이 될 것이다.

참고문헌

[1] G. Leen and D. Herrernam, "Expending automotive electronic" *IEEE Computer,* vol. 35, pp. 88-93, 2002.

[2] J. Ryu, M. Yoon, and M. Sunwoo, "Development of a network-based traction control system, validation of its traction control algorithm and evaluation of its performance using Net-HILS" *Int. J. Automotive Technology*, vol. 7, no. 6, pp. 687-695, 2006.

[3] H. Kopetz, "A Comparison of CAN and TTP," *Annual Reviews in Control*, vol. 24, no. 1, pp. 177-188, 2000.

[4] 오원현, 권형근, 김정찬, 윤형진, "모델기반 자동코드생성을 이용한 전동기 구동 시뮬레이터 개발," 한국자동차공학회 춘계학술대회 논문집, pp. 1607-1612, 2004.

[5] 김만호, 하경남, 이석, 이경창, "노드 기반 스케줄링 방법을 이용한 FlexRay 네트워크 시스템의 구현," *Transactions of KSAE*, vol. 18, no. 2, pp. 39-47, 2010

[6] FlexRay Consortium, FlexRay Protocol Specification V2.1, Revision A, 2005.

윤 승 현

2008년 성균관대학교 정보통신공학부 졸업. 2008년~현재 성균관대학교 전자전기컴퓨터공학과 석사과정 재학 중. 관심분야는 임베디드 시스템.

황 성 호

1988년 서울대학교 기계설계학과 졸업. 1990년 동 대학원 석사. 1997년 동 대학원 박사. 1992년~2002년 한국생산기술연구원 선임연구원. 2002년~현재 성균관대학교 기계공학부 부교수. 관심분야는 자동차 메카트로닉스 하이브리드 전기자동차, 유공압 제어시스템.

전 재 욱

1984년 서울대학교 전자공학과 졸업. 1986년 동 대학원 석사. 1990년 Purdue University 전기공학과 박사. 1990년~1994년 삼성전자 생산기술센터 선임연구원. 1994년~현재 성균관대학교 전자전기컴퓨터공학부 교수. 관심분야는 로봇공학, 마이크로프로세서.

서 석 현

2005년 성균관대학교 전자전기컴퓨터공학과 졸업. 2007년 동 대학원 석사. 2007년~현재 동 대학원 박사과정 재학 중. 관심분야는 ECU, 임베디드 시스템.

권 기 호

1975년 성균관대학교 전자공학과 졸업. 1978년 서울대학교 전자공학과 석사. 1978년~1980년 ETRI 연구원. 1988년 서울대학교 전자공학과 박사. 1989년~현재 성균관대학교 전자전기컴퓨터공학부 교수. 관심분야는 퍼지 시스템, 유전자 알고리즘, 카오스이론.