Transactions of KSAE, Vol. 13, No. 6, pp.48-55 (2005)

1225-6382/2005/078-07

LIN 프로토콜 시간 모델링 및 메시지 응답 시간 해석에 관한 연구

연 제 명[1] · 선 우 명 호[*1] · 이 우 택[2]

한양대학교 자동차공학과[1] · 창원대학교 메카트로닉스공학부[2]

A Study on Timing Modeling and Response Time Analysis in LIN Based Network System

Jeamyoung Youn[1] · Myoungho Sunwoo[*1] · Wootaik Lee[2]

[1]Department of Automotive Engineering, Hanyang University, Seoul 133-791, Korea
[2]Department of Mechatronics Engineering, Changwon National University, Gyeongnam 641-773, Korea
(Received 3 September 2005 / Accepted 22 July 2005)

Abstract : In this paper, a mathematical model and a simulation method for the response time analysis of Local Interconnect Network(LIN) based network systems are proposed. Network-induced delays in a network based control system can vary widely according to the transmission time of message and the overhead time of transmission. Therefore, in order to design a distributed control system using LIN network, a method to predict and verify the timing behavior of LIN protocol is required at the network design phase. Furthermore, a simulation environment based on a timing model of LIN protocol is beneficial to predict the timing behavior of LIN. The model equation is formulated with six timing parameters deduced from timing properties of LIN specification. Additionally, LIN conformance test equations to verify LIN device driver are derived with timing constraints of the parameters. The proposed model equation and simulation method are validated with a result that is measured at real LIN based network system.

Key words : LIN(Local Interconnect Network), Response time(응답시간), Network(네트워크)

1. 서 론

현재 자동차에는 다양한 네트워크 프로토콜이 적용되고 있으며, 적용 분야와 전송 속도에 따라 네트워크를 Class A, B, C로 분류하고 있다.[1] Local Interconnect Network(LIN)은 Class A에 속하는 저속 통신 네트워크 프로토콜로 스마트 액추에이터와 스마트 센서를 포함하는 차량 문, 실내 공조, 좌석, 선루프 등의 제어 장치에 적용을 목적으로 제안되었다.[2-4]

LIN 네트워크를 기반으로 하는 자동차 전자제어 시스템(ECU)은 응답 시간에 대한 제약을 가지는 실시간 시스템으로, 다수의 ECU가 독립적으로 개발·운영되는 분산 제어 방식을 따른다. 실시간 분산 제어 시스템의 중요한 성능 평가 지수인 응답 시간은 제어기의 응답 시간과 네트워크의 응답 시간으로 구성된다. 따라서 설계 단계에서 LIN 네트워크에 의하여 발생하는 시간 지연을 고려하여 제어 시스템을 설계할 필요가 있다.[5-7]

네트워크의 응답 시간 해석 방법에는 네트워크의 절대적인 안정을 보장하기 위하여, 발생 가능한 최악의 통신 상황을 고려하는 최악 응답 시간 해석 방법이 있다. 최악 응답 시간은 네트워크가 최악의 상황에 놓였을 때의 메시지의 응답 시간을 의미하므로, 수학적으로 표현하여 계산하는 것이 비교적 용

*To whom correspondence should be addressed.
msunwoo@hanyang.ac.kr

이하다. 하지만, 정상적인 네트워크 상태에서 최악의 통신 상황이 발생할 확률은 높지 않으며, 컴퓨터 시뮬레이션을 이용한 네트워크 응답 시간 해석에 단순한 최악 응답 시간 모델을 이용하는 것은 적합하지 않다.[8] 따라서 보다 현실적인 LIN 네트워크의 응답 시간의 해석을 위해서는 LIN 네트워크의 특성을 고려한 네트워크 응답 시간 모델을 기반으로 하는 컴퓨터 시뮬레이션 환경의 개발이 필요하다. Chen[9]은 LIN 네트워크 마스터 노드와 슬레이브 노드로 나누고, 각 노드의 구성 요소를 하드웨어와 소프트웨어로 구분지어 모델링을 수행하였으며, LIN 네트워크에서 발생할 수 있는 다양한 상황을 시뮬레이션 할 수 있는 환경을 개발하였다.

이 연구에서는 LIN 네트워크의 응답 시간에 영향을 주는 요소를 메시지 스케줄링과 메시지 전송 시간으로 나누어 모델링을 수행하였다. 메시지 스케줄링에 의한 영향을 고려하기 위한 시뮬레이션 의사코드(pseudo-code)와, 전송 시간에 대한 수학적 모델링 방법과 모델 식을 제시하였으며, 최종적으로 위의 두 요소를 고려하여 LIN 네트워크의 응답 시간 해석을 위한 시뮬레이션 환경을 구성하고, 실제 LIN 네트워크 시스템에 대한 시뮬레이션을 수행하여 타당성을 검증하였다.

2. LIN 프로토콜의 특성

2.1 기본 특성

LIN은 가격과 안정성에 중점을 두어 설계된 프로토콜이다. 저가의 통신 환경을 구성하기 위하여 단선과 간단한 발진기를 사용하도록 하였으며, 범용 직렬통신을 기반으로 통신을 구현하였다. 네트워크의 안정성을 확보하기 위하여 12[V] 버스, 다양한 오류 검출 알고리즘, 동기화만을 위한 메시지 필드 할당 등을 적용하고 있다. 또한 마스터/슬레이브(master/slave) 방식의 메시지 전송 방식을 사용함으로써 네트워크 관리를 단순화 하였으며, 정확한 응답 시간의 예측을 가능하게 하였다.[3,4]

2.2 메시지 프레임의 구성과 전송 방식

LIN의 메시지 프레임은 헤더(header) 프레임과 응답(response) 프레임으로 구성 된다. Fig. 1은 LIN 메시지 프레임의 구성을 보여주고 있다. 헤더 프레임은 동기화 시작(synchbreak) 필드, 동기화(synch) 필드와 메시지 아이디 필드로 구성 되며, 응답 프레임은 2, 4, 8 바이트의 데이터 필드와 데이터 전송 오류 확인을 위한 체크섬(checksum) 필드로 이루어진다.

Fig. 2는 LIN 네트워크의 다양한 메시지 전송방식을 도시한 것이다. 마스터 태스크는 마스터 노드에만 존재하는 태스크로, 헤더 프레임의 전송 순서와 시기를 결정하며, 네트워크 슬립과 웨이크업을 동기화하는 역할을 수행한다. 슬레이브 태스크는 마스터 노드를 포함한 모든 노드에 존재하는 태스크로, 헤더 프레임의 메시지 아이디에 따라 응답 프레임을 송·수신하는 역할을 수행한다.

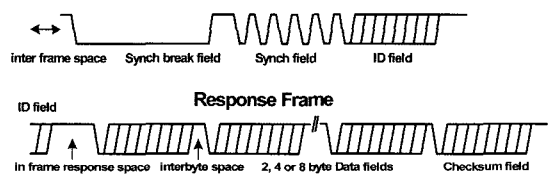
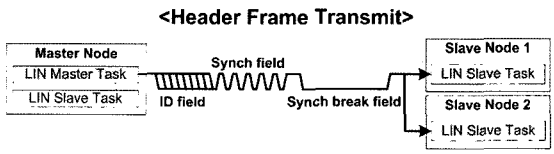
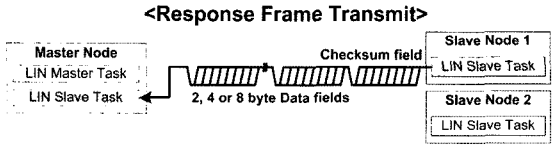


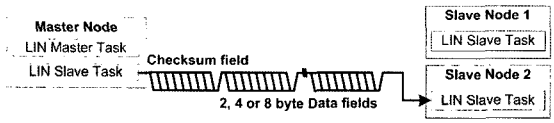
Fig. 1 The structure of a LIN frame



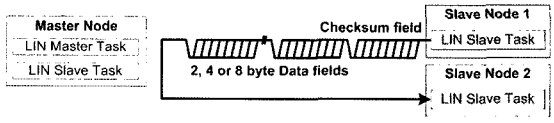
(a) Transmitting header from the master to salves



(b) Transmitting data from a slave to the master



(c) Transmitting data from the master to a slave or slaves



(d) Transmitting data from a slave to a slave or slaves

Fig. 2 LIN communication method

3. LIN 프로토콜 응답 시간 모델링

LIN 네트워크의 모든 메시지가 주기적으로 전송되고 전송 주기와 마감 시간이 같은 경우, LIN 메시지 m의 시간 속성은 (P_m, C_m)과 같이 표현 할 수 있다.[10] 여기서, P_m은 메시지의 전송 주기이고, C_m은 메시지 전송 시간이다.

3.1 메시지 스케줄링 모델링

LIN 프로토콜에서는 Fig. 3과 같이 헤더 프레임의 전송을 위하여 메시지의 전송 주기와 전송 시간에 기초하여 스케줄 표를 오프라인에서 작성하는 정적(static) 스케줄링 방식을 사용하도록 하고 있다.[4] 정적 스케줄링 방식은 스케줄가능여부(schedulability)를 설계 단계에서 보장 할 수 있는 장점을 가지며, 스케줄링을 위한 제어기의 계산 요구량이 적어 성능이 낮은 제어기에서 구동하기에 유리하다.[11,12] 또한, LIN 프로토콜에서는 정적 스케줄링 방식의 단점인 유연성(flexibility)을 보완하기 위하여 제어 상황을 고려하여 여러 개의 스케줄 표를 작성하고, 제어기의 동작 상황에 따라 스케줄 표를 변경하는 모드 변환 방식을 사용하도록 하고 있다.[4]

정적 스케줄링 방식에 따라 메시지 전송을 위한 스케줄 표를 작성할 때에, 마감 시간이 전송 주기와 동일하다고 가정하면, 메시지 m의 최악 응답 시간은 메시지의 주기 P_m과 같다.

메시지의 응답 시간 해석을 위해서는, Fig. 3과 같이 작성된 스케줄 표의 시간 거동을 모사하는 시뮬레이션을 수행하여야 한다. 정적 스케줄 방식의 시뮬레이션을 위한 의사코드는 Fig. 4와 같다.[13] 여기서, H는 대주기(hyperperiod)이고, N은 H 동안 전송되는 메시지의 개수 이다. H는 전체 메시지의 전송 주기의 최소 공배수 값으로 결정한다. 표에 존재하는 각 메시지의 시간 속성은 $(t_k, m(t_k))$로 표현하는데, t_k는 k번째 메시지의 전송 시점을 의미하며, $m(t_k)$은 t_k 시점에서 전송되는 메시지를 가리킨다.

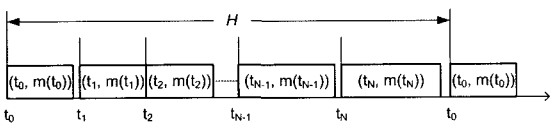


Fig. 3 A schedule table in a master node

```
Message SCHEDULER:
  set the next decision point i and table entry k to 0;
  set the timer to expire at t_k;
  do forever:
    current message m = m(t_k);
    increment i by 1;
    compute the next table entry k = i mod (N);
    set the timer to expire at ⌊i/N⌋H + t_k;
    sleep
end SCHEDULER
```

Fig. 4 Pseudocode of a static scheduling kernel

3.2 메시지 응답 시간 모델링

메시지 전송을 위한 스케줄 표의 작성과 응답 시간 해석을 위해서는 메시지 전송 시간에 대한 모델이 요구된다. 전송 시간을 모델링하기 위해서는 메시지를 전송하는 LIN 네트워크 디바이스 드라이버(device driver)를 고려하여 전송 시간에 영향을 주는 요인을 분석하고, 메시지 전송 시간을 수학적으로 표현하여야 한다. LIN 네트워크의 슬립과 웨이크업 상황을 제외하면, LIN 메시지 프레임은 헤더 프레임과 응답 프레임으로만 구성되며, 메시지 전송 시간을 식 (1)과 같이 두 프레임의 전송 시간의 합으로 볼 수 있다.[3]

$$C_m = T_{frame} = T_{header} + T_{response} \tag{1}$$

여기서, T_{header}는 헤더 프레임의 전송 시간이며, $T_{response}$는 응답 프레임의 전송 시간이다.

3.2.1 헤더 프레임의 전송 시간: T_{header}

헤더 프레임의 전송 시간에 포함되는 헤더 프레임의 전송 준비 시간은 동기화 시작 필드의 전송을 준비하는 데에 소요되는 시간이다. 이 시간은 메시지 전송 속도나 메시지 길이에 상관없이 일정한 값을 가지므로, 한 개의 시간 상수 P_{INTER}로 표현 할 수 있다.

동기화 시작 필드는 메시지의 전송 시작을 알리는 필드로 슬레이브 노드가 도미넌트(dominant: 0[V]의 전압 상태) 상태의 11비트 값을 인식할 수 있도록 최소 $13\tau_{bit}$의 시간 동안 버스를 도미넌트 상태로 유지하여야 한다. $13\tau_{bit}$의 시간은 슬레이브 노드에서 사용하는 RC 오실레이터의 오차 범위를 최대 ±15%까지 허용하기 위한 최소값이다. 동기화 시작 필드의 종료를 위해서는 도미넌트 상태 이후에 최

소 $1\tau_{bit}$의 시간 동안 버스를 리세시브(recessive: 12[V]의 전압 상태) 상태로 유지하여야 한다.[14] Fig. 5는 동기화 시작 필드의 버스 상태를 도시한 것이다. 동기화 시작 필드에서 T_{SYNBRK}는 도미넌트 상태의 버스 시간을 의미하며, T_{SYNDEL}는 리세시브 상태의 버스 시간을 의미한다. T_{SYNBRK}에서 $13\tau_{bit}$을 초과하는 시간과 T_{SYNDEL}에서 $1\tau_{bit}$을 초과하는 시간은 P_{SYNBRK}와 P_{SYNDEL}의 두 개의 시간 파라미터로 표현한다. 동기화 시작 필드는 LIN 디바이스 드라이버의 설계 단계에서 일정한 길이로 고정되므로 두 시간 파라미터는 상수가 된다. 따라서 동기화 시작 필드는 식 (2)와 같이 표현 된다.

$$T_{synch\_break} = 14\tau_{bit} + P_{SYNBRK} + P_{SYNDEL} \quad (2)$$

동기화 필드와 아이디 필드는 Fig. 6과 같이 8N1 바이트 형식에 따라 전송되므로, 식 (3)과 같이 표현된다.

$$T_{synch\_field} = T_{identifier\_field} = (8+2)\tau_{bit} \quad (3)$$

동기화 필드와 아이디 필드 사이에서 발생하는 시간 지연은 메시지 아이디의 전송을 준비하는데 사용되는 시간으로, LIN 소프트웨어의 설계에 의하여 일정한 시간을 소요하므로 한 개의 시간 상수 P_{ID}로 고려한다.

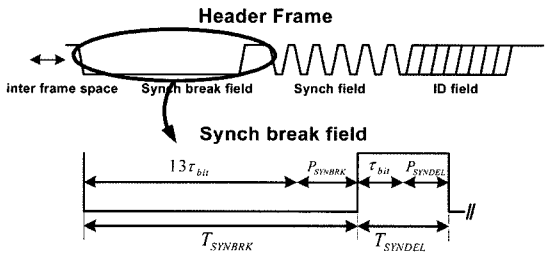


Fig. 5 Synch break field

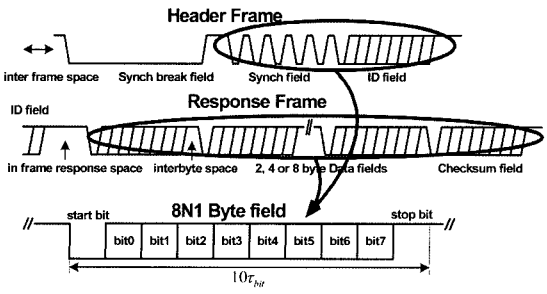


Fig. 6 Byte format of 8N1

따라서 헤더 프레임의 전송 시간은 식 (4)와 같이 네 개의 시간 파라미터를 가지는 형태로 표현 된다.

$$T_{header} = 34\tau_{bit} + P_{INTER} + P_{SYNBRK} + P_{SYNDEL} + P_{ID} \quad (4)$$

3.2.2 응답 프레임의 전송 시간: $T_{response}$

응답 프레임은 헤더 프레임의 아이디 값에 따라 해당하는 한 개의 슬레이브 태스크에서 전송되며, 데이터 필드와 데이터의 전송 오류를 확인하기 위한 체크섬 필드로 구성된다. 응답 프레임을 송신 하는 노드는 헤더 프레임을 수신한 이후에 필요한 데이터의 전송을 준비한다. 이와 같은 동작에 소요되는 시간을 응답 준비 시간이라 한다. 이 시간은 아이디 검색 시간과 데이터 준비 시간으로 구성 되는데, LIN의 경우 노드에 할당 되는 아이디의 개수가 많지 않으므로 아이디 검색 시간은 무시할 수 있다. 데이터 준비 시간은 전송하는 데이터 크기 s_m에 비례하여 증가하는 특성을 가진다. 따라서 응답 준비 시간($P_{INFRAME}$)은 식 (5)와 같이 전송 데이터의 길이에 비례하여 증가하는 일차 선형식으로 표현할 수 있으며, 일차식의 두 계수는 P_{IF_1}, P_{IF_2}로 표기한다.

$$P_{INFRAME} = P_{IF\_1} s_m + P_{IF\_2} \quad (5)$$

데이터 필드와 체크섬 필드는 8N1 바이트 형식에 따라 전송되므로 식 (3)과 동일하게 표현된다. 전송되는 각 데이터 바이트 사이에서 발생하는 시간 지연인 데이터 바이트 간격(interbyte space)은 데이터의 크기와 전송 속도에 관계없이 일정한 값을 가지므로, 한 개의 시간 상수 $P_{INTERBYTE}$로 고려하도록 한다. 따라서 $T_{response}$는 다음 식 (6)과 같이 정의된다.

$$T_{response} = 10(s_m + 1)\tau_{bit} + P_{INFRAME} + P_{INTERBYTE}\, s_m \quad (6)$$

응답 프레임의 전송 시간은 식 (6)과 같이 두 개의 시간 상수와 전송 속도, 데이터 크기의 관계식으로 표현된다.

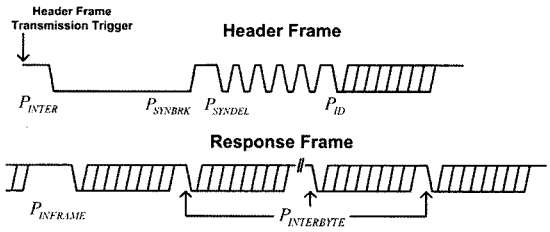


Fig. 7 Message frame with time parameters

3.2.3 메시지 전송 시간: C_m

메시지 전송 시간에 대한 식은 식 (4)와 식 (6)을 이용하여 식 (7)과 같이 정의 된다.

$$C_m = (44 + 10s_m)\tau_{bit} + P_{INTER} + P_{SYNBRK} + P_{SYNDEL} \\ P_{ID} + P_{INFRAME} + P_{INTERBYTE}s_m \quad (7)$$

결과적으로 메시지의 전송 시간은 앞서 정의한 여섯 개의 시간 파라미터와 데이터의 길이, 비트 전송시간의 관계식으로 정의된다. 여섯 개의 시간 파라미터는 LIN 디바이스 드라이버의 시간 특성을 분석함으로써 구할 수 있다.

3.2.4 최악 메시지 전송 시간

LIN 프로토콜에서는 메시지 전송 시간에 대한 최대 허용 값을 식 (8)과 같이 최소 메시지 전송 시간의 140%로 정의하고 있다.[3)]

$$[C_{m_A}]_{\max} = [C_{m_A}]_{\min} \times 1.4 \quad (8)$$

따라서 LIN 메시지의 최악 전송 시간은 $[C_{m_A}]_{\min} \times 1.4$ 이 되며, 최악의 경우를 고려한 응답 시간의 해석에 사용된다.

3.2.5 시간 파라미터를 이용한 LIN 디바이스 드라이버의 성능 평가

식 (4)의 헤더 프레임 전송 시간 모델에 사용된 파라미터들의 최대값은 Table 1에 제시된 헤더 프레임의 최대 전송 시간을 이용하여 식 (9)와 같이 구할 수 있다.

$$[P_{INTER} + P_{SYNBRK} + P_{SYNDEL} + P_{ID}]_{\max} = 15\tau_{bit} \quad (9)$$

마스터 태스크를 구동하는 LIN 디바이스 드라이버는 드라이버의 시간 파라미터들을 이용하여 구한 값이 식 (9)의 최대 허용 값을 넘지 않도록 설계되어야 한다.

식 (6)의 응답 프레임 전송 시간 모델에 사용된 파라미터들의 최대값은 Table 1에 제시된 응답 프레임의 최대 전송 시간을 이용하여 식 (10)과 같이 구할 수 있다.

Table 1 Time constraint of message frame

Frame	Maximum transmission time
$[T_{header}]_{\max}$	$([T_{header}]_{\min} + 1) \times 1.4 = 49\tau_{bit}$
$[T_{response}]_{\max}$	$([T_{response}]_{\min} + 1) \times 1.4 = 14(1 + s_m)\tau_{bit}$

$$[P_{INFRAME} + P_{INTERBYTE}s_m]_{\max} = 4(s_m + 1)\tau_{bit} \quad (10)$$

슬레이브 태스크를 구동하는 LIN 디바이스 드라이버는 드라이버의 시간 파라미터들을 이용하여 구한 값이 식 (10)의 최대 허용 값을 넘지 않도록 설계되어야 한다.

따라서 LIN 디바이스 드라이버의 성능은 마스터 태스크와 슬레이브 태스크 각각에 대하여 식 (9)와 식 (10)을 이용하여 평가할 수 있다. LIN 디바이스 드라이버의 시간 파라미터 값을 이용하여 구한 식 (9)와 식 (10)의 값이 작을수록 전송 시간에 의한 시간 지연이 줄어들어 메시지 응답 시간이 짧아진다.

4. 메시지 응답 시간 해석

메시지 응답 시간 해석을 위하여 Fig. 8과 같이 마스터 노드를 포함한 세 개의 노드로 LIN 네트워크 시스템을 구성한다. 각 노드의 ECU로는 프리스케일(*freescale*)사의 HC12D60 마이크로컨트롤러를 사용하며, 메시지의 송·수신을 위하여 프리스케일에서 제공하는 HC12Dxx 계열용 디바이스 드라이버를 사용한다.[15)]

4.1 전송 시간 모델의 시간 파라미터 측정

LIN 프로토콜의 시간 모델에 사용되는 시간 파라미터들은 ECU가 LIN 디바이스 드라이버를 구동하는데 소요하는 실행 시간을 측정하여 계산 할 수 있다. 드라이버의 코드 실행 시간은 인터럽트나 다른 소프트웨어에 의하여 코드의 실행이 영향을 받지 않는 조건하에 측정하여야 한다.[16,17)] 이 연구에서 사용하는 모든 ECU는 LIN 디바이스 드라이버 이외의 다른 소프트웨어나 인터럽트를 사용하지 않도록 구성하였으므로, LIN 버스 시그널을 이용하여 소프트웨어의 실행 시간을 측정할 수 있다.

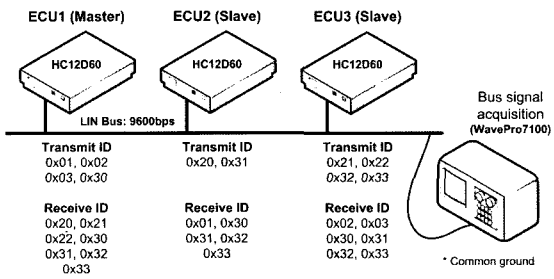


Fig. 8 LIN based communication system for test

프리스케일사의 LIN 디바이스 드라이버의 시간 파라미터를 찾기 위하여 메시지 전송 속도를 9600bps로 고정하고 응답 노드와 전송 데이터의 길이를 Table 2, 3, 4와 같이 변경하면서, 각 조건에 대하여 10회 반복하여 LIN 버스 시그널을 측정하였다. 측정 결과를 이용하여 마스터 노드의 마스터 태스크와 마스터 노드와 슬레이브 노드의 슬레이브 태스크에 대한 시간 파라미터를 계산하였다. 측정된 모든 시간파라미터 값의 표준 편차는 $1\mu s$로 이하로 나타났으며, 이 결과는 LIN 버스 시그널을 이용한 시간 파라미터 계산 방법이 유용하다는 것을 의미한다. 측정된 *freescale*의 LIN 디바이스 드라이버의 시간 파라미터를 이용하여 계산한 식 (9)와 식 (10)의 드라이버 성능 평가 결과는 Table 5와 같다. Table 5의 결과는 *freescale*에서 제공하는 LIN 디바이스 드라이버가 LIN 프로토콜에서 제시하는 시간 성능을 만족함을 보여준다.

Table 2 Parameter of header frame

Response Node	P_{INTER}	P_{SYNBRK}	P_{SYNDEL}	P_{ID}
Master	$178\mu s$	$49\mu s$	$170\mu s$	$116\mu s$
Slave	$178\mu s$	$49\mu s$	$170\mu s$	$116\mu s$
Average	$178\mu s$	$49\mu s$	$170\mu s$	$116\mu s$
Standard deviation	$0.76\mu s$	$0.46\mu s$	$0.50\mu s$	$0.10\mu s$

Table 3 Parameter of response frame in master node

s_m	$P_{INFRAME}$	$P_{INTERBYTE}$
2	$135\mu s$	$123\mu s$
4	$142\mu s$	$121\mu s$
8	$154\mu s$	$122\mu s$
2	$135\mu s$	$123\mu s$

P_{IF_1}	P_{IF_2}	$P_{INTERBYTE}$
$3.3\mu s$	$128\mu s$	Average: $122\mu s$ Standard deviation: $0.76\mu s$

Table 4 Parameter of response frame in slave node

s_m	$P_{INFRAME}$	$P_{INTERBYTE}$
2	$137\mu s$	$122\mu s$
4	$144\mu s$	$121\mu s$
8	$159\mu s$	$122\mu s$
2	$137\mu s$	$122\mu s$

P_{IF_1}	P_{IF_2}	$P_{INTERBYTE}$
$3.6\mu s$	$130\mu s$	Average: $122\mu s$ Standard deviation: $0.48\mu s$

Table 5 Conformance of LIN device driver by *freescale*

Task	Node	Eq. (9) and Eq. (10)	Maximum transmission time
Master	Master	335×10^{-6}	1562×10^{-6}
Slave	Master	$(128+125.3s_m)10^{-6}$	$(416+416s_m)10^{-6}$
	Slave	$(130+125.6s_m)10^{-6}$	$(416+416s_m)10^{-6}$

4.2 메시지 스케줄링

LIN 네트워크 시스템의 메시지 데이터베이스가 Table 6과 같을 때, 메시지의 전송 주기와 전송 시간을 고려하여 작성한 마스터 태스크의 스케줄 표는 Fig. 9와 같다. 여기서, ECU1이 마스터 노드의 역할을 수행하며, 나머지 ECU2와 ECU3이 슬레이브 노드로 동작한다. 스케줄 표 작성에 이용되는 전송 시간은 앞서 제시한 식 (7)의 전송 시간 모델식을 이용한다. Table 7에서 Model 부분은 식 (7)로부터 이론적으로 구한 값이며, Measure 부분은 버스 신호를 직접 측정한 값이다. 두 데이터 간의 오차 범위가 수십μ초 이내로 나타나는 것을 확인할 수 있다. 또한 Worst 부분의 최악 전송시간이 실제 전송시간에 비

Table 6 LIN message database

ID	s_m	Period	Tx	Rx	ID	s_m	Period	Tx	Rx
0x01	2	100ms	ECU1	ECU2	0x22	4	200ms	ECU3	ECU1
0x02	2	100ms	ECU1	ECU3	0x30	8	400ms	ECU1	All
0x03	2	100ms	ECU1	ECU3	0x31	8	400ms	ECU2	All
0x20	4	200ms	ECU2	ECU1	0x32	8	400ms	ECU3	All
0x21	4	200ms	ECU3	ECU1	0x33	8	400ms	ECU3	All

Table 7 Message transmission time

ID	Model	Measure	Worst	ID	Model	Measure	Worst
0x01	7.558ms	7.56ms	9.583ms	0x22	9.895ms	9.89ms	12.5ms
0x02	7.558ms	7.55ms	9.583ms	0x30	14.560ms	14.56ms	18.3ms
0x03	7.558ms	7.56ms	9.583ms	0x31	14.564ms	14.57ms	18.3ms
0x20	9.895ms	9.90ms	12.5ms	0x32	14.564ms	14.56ms	18.3ms
0x21	9.895ms	9.89ms	12.5ms	0x33	14.564ms	14.56ms	18.3ms

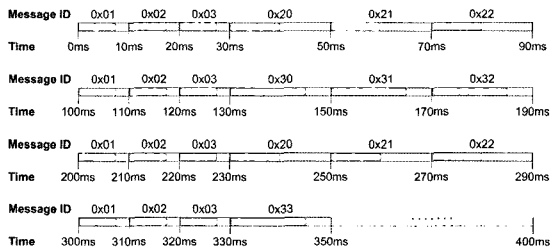


Fig. 9 The schedule table of header frame transmission

하여 125%정도 크게 계산되는 것을 볼 수 있다. 결과적으로, 메시지 스케줄 표를 작성할 때에 최악 전송시간을 이용하는 방법보다는 모델을 이용하여 구한 전송시간을 이용하는 것이 유용하다고 할 수 있다.

4.3 메시지 응답 시간 시뮬레이션

LIN 네트워크의 메시지 응답 시간 해석을 위하여 *Mathworks*의 Matlab을 이용하여, 앞서 제시한 메시지 전송 스케줄의 모사를 위한 의사코드와 메시지 전송 시간 모델을 기반으로 시뮬레이션을 수행하였으며, 그 결과는 Fig. 10과 같다.

시뮬레이션 결과를 검증하기 위하여 스케줄 표에 따라 메시지를 전송하도록 마스터 태스크를 설정하고, 데이터베이스에의 설정에 따라 세 개의 노드가 동작하도록 ECU에 프로그램 하여 측정한 LIN 버스 시그널은 Fig. 11과 같다. 실제 측정한 LIN 메시지의 응답 시간과 시뮬레이션을 통하여 예측된 응답 시간의 결과가 일치하는 것을 볼 수 있다. Fig. 12는 시뮬레이션 결과를 보다 자세히 비교하기 위하여, 0×33 메시지를 확대하여 도시한 것이다. 그림에서와 같이 메시지의 각 필드 수준까지 응답 시간을 정확하게 예측할 수 있음을 알 수 있다.

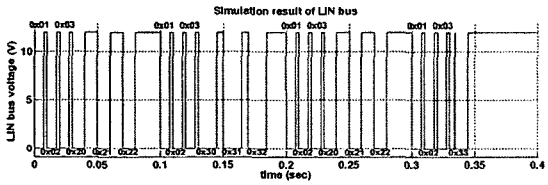


Fig. 10 Simulation result of LIN message response time

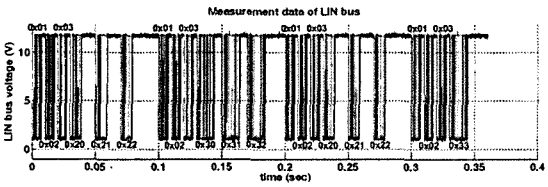


Fig. 11 Measurement result of LIN bus

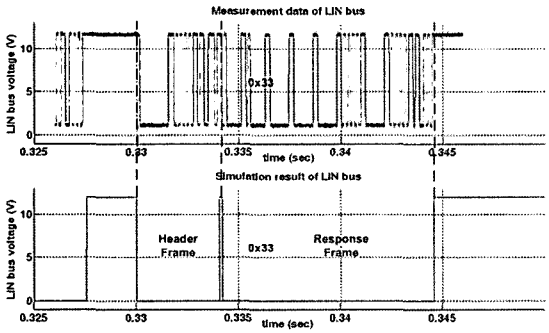


Fig. 12 Measurement and simulation result of ID 0×33

5. 결 론

이 연구에서는 저속 네트워크 표준으로 제안된 LIN 네트워크 시스템의 응답 시간 해석을 위한 모델링 방법과 모델 식을 제시하였다. 또한, LIN 네트워크를 마스터 태스크의 메시지 전송 스케줄러와 메시지 전송 시간으로 나누어 모델링을 수행하였다. 스케줄러는 의사코드로 모델링 하였으며, 메시지 전송 시간은 LIN 프로토콜의 시간 특성을 고려하여 여섯 개의 시간 파라미터를 이용하여 모델링 하였다. 제시된 모델을 바탕으로 LIN 네트워크의 응답 시간을 해석하기 위한 시뮬레이션 환경을 구성하였으며, 실제 네트워크 시스템과 비교하여 실효성을 검증하였다. 추가적으로, 여섯 개의 시간 파라미터를 사용하여 LIN 디바이스 드라이버의 성능을 평가하기 위한 방법을 제시하였다.

이 연구를 바탕으로 LIN을 포함한 차량 네트워크 시스템의 응답 시간을 예측 할 수 있는 환경을 구성할 수 있을 것으로 기대 되며, 이를 기반으로 네트워크 기반의 제어 시스템의 응답 시간을 최적화는 연구가 진행 될 수 있을 것이다.

후 기

이 연구는 과학기술부의 국가지정연구실사업(NRL)의 지원에 의하여 수행되었다.

References

1) C. A. Lupini, "Multiplex Bus Progression," SAE In-Vehicle Networks, 2001-01-0060, 2001.
2) J. V. Denuto, S. Ewbank, F. Kleja, C. A. Lupini and R. A. Perisho, "LIN Bus and its Potential for Use in Distributed Multiplex Applications,"

SAE In-Vehicle Networks, 2001-01-0072, 2001.
3) LIN Specification, Revision 1.2, Nov. 17, 2002.
4) LIN Specification, Revision 2.0, Sep. 23, 2003.
5) M. S. Shin, W. T. Lee and M. H. Sunwoo, "Holistic Scheduling Analysis of a Can Based Body Network System," Transactions of KSAE, Vol.10, No.5, pp.114-120, 2002.
6) K. Tindell and J. Clark, "Holistic Schedulability Analysis for Distributed Hard Real-time Systems," Microprocessors and Microprogramming, pp.117- 134, 1994.
7) M. H. Sunwoo, S. M. Shin, W. T. Lee and S. Y. Han, "Development of a Body Network System with OSEK/VDX Standards and CAN Protocol," Transactions of KSAE, Vol.10, No.4, pp.175-180, 2002.
8) J. Sun and J. Liu, "Synchronization protocols in Distributed Real-time Systems," 16th IEEE International Conference on Distributed Computing Systems, pp.38-45, 1996.
9) Q. Chen, Y. Dong and S. Momin, "Cycle Accurate LIN Network Modeling and Simulation," SAE In-Vehicle Networks (SP-1658), pp.113- 117, 2002.
10) K. Tindell and A. Burns, "Guaranteeing Message Latencies on Control Area Network (CAN)," Technical Report, Department of Computer Science, University of York, England, 1994.
11) L. Almeida, R. Pasadas and J. A. Fonseca, "Using the Planning Scheduler in Real-time Fieldbuses: Theoretical Model for Run-time Overhead," IEEE International Workshop on Factory Communication Systems, pp.103-109, 1997.
12) T. P. Baker and A. Shaw, "The Cyclic Executive Model and Ada," Real-Time Systems Symposium, pp.120-129, 1988.
13) J. Liu, Real-time Systems, Prentice Hall, pp.85-87, 2000.
14) J. W. Specks and A. Rajnák, "LIN - protocol, Development Tools, and Software Interfaces for Local Interconnect Networks in Vehicles," VDI BERICHTE, Vol.1547, pp.227-250, 2000.
15) Freescale Semiconductor, http://www.freescale.com
16) M. Lindgren, H. Hansson and H. Thane, "Using Measurements to Derive the Worst-case Execution Time," 17th IEEE International Conference on Real-Time Computing Systems and Applications, pp.15-22, 2000.
17) W. Zhao, P. Kulkarni, D. Whalley, C. Healy, F. Mueller and G. R. Uh, "Tuning the WCET of Embedded Applications," 10th IEEE Real-Time and Embedded Technology and Applications Symposium, 2004.