Transactions of the Korean Society of Automotive Engineers, Vol.11, No.1, 2003

분산형 실시간 제어시스템을 위한 연계 모의실험에 관한 연구

김 승 훈* · 이 우 택 · 선 우 명 호

한양대학교 자동차공학과

A Co-simulation Toolbox for Distributed Real-Time Control System

Seunghoon Kim* · Wootaik Lee · Myoungho Sunwoo

Department of Automotive Engineering, Hanyang University, Seoul 133-791, Korea

(Received 6 August 2002 / Accepted 29 October 2002)

Abstract : This paper presents the algorithms and Matlab Toolbox for co-simulation of distributed real-time control system based on OSEK-OS and CAN protocol.This toolbox enables the developers to analyze the timing uncertainty, which is caused by resource sharing including shared memories and networks, and to take the timing uncertainty into consideration in the early design phase. Furthermore, this toolbox helps the developers to model the behaviors of a control system by providing graphical user interface for objects of OSEK-OS and CAN. To prove the feasibility of this toolbox, a vehicle body network system is modeled with this toolbox, and the timing uncertainties are analyzed.

Key words : Co-simulation(연계 모의실험), RTOS(Real-Time Operating System;실시간 운영체제), Execution time(실행시간), OSEK-OS, CAN(Control Area Network), Scheduling(스케쥴링)

1. 서 론

최근 산업 전반에 걸쳐 분산형 실시간 제어시스템에 대한 관심이 높아지면서 이에 대한 활발한 연구가 진행되고 있다. 또한 객체지향적인 개발 방법을 적용하기 위하여 실시간 운영체제를 도입하는 사례가 증가하게 되었고, 이는 소프트웨어적인 측면에서 생산성을 향상시켰으나 제어 태스크(Task)의 실행시간과 태스크의 스케줄링으로 인하여시간 불확실성을 야기하게 된다. 특히 분산형 실시간 제어시스템의 경우는 각 제어기가 공유하는 네트워크의 스케줄링으로 인하여 시간 불확실성이 더욱 커지게 된다. 이러한 시간 불확실성은 제어 성능을 떨어뜨릴 뿐만 아니라, 경우에 따라서는 제어시스템에 요구되는 실시간 성능을 보장할 수 없게 된다.[1)]

분산형 실시간 제어시스템에서 시간 불확실성을 해석하고실시간 성능을 검증하는 것은 매우 중요하다. 이러한 시간 불확실성의 문제를 설계 초기부터 고려하여 제어기를 설계하는 것이 바람직하다. 그러나, 기존에는 시간문제를 고려하여 평가할 수 있는 모의실험 환경이 없었으므로 하드웨어를 구성한 후 최종적인 성능 검증 단계에서나 가능하였다.

최근 Eker와 Cervin (1999) 이 연계 모의실험을 가능하게 하는 알고리즘을 연구하고 그 결과로 Matlab에서 사용할 수 있는 Toolbox를 개발하였다.[2)] 그러나 이 Toolbox는 일반적인 실시간 운영체제를 가정하고 설계되어 실제 운영체제에서 사용하는 다양한 기능들을 모사하지 못하며, 네트워크 모델링을 단순화하여 실제의 분산형 실시간 제어시스템의 설계에 적용이 난해하다. 또한, 일정 시간 간격으로만 모

*To whom correspondence should be addressed.
ksh819@orgio.net

의실험할 수 있도록 구성되어 계산효율이 좋지 못하다.

이 연구에서는 최근 자동차 산업에서 실시간 운영체제의 표준으로 자리 잡아가는 OSEK-OS[3]와 제어 네트워크로 널리 사용되는 CAN[4]을 사용한 분산형 실시간 제어시스템의 시간 불확실성을 정밀하게 해석할 수 있는 연계 모의실험 알고리즘에 대하여 연구하고 설계시 효율적으로 사용할 수 있는 Matlab의 Toolbox를 제시한다. 또한 이 Toolbox를 이용하여 차체 네트워크 시스템을 모델링하고 시간 불확실성을 해석한다.

2. 연계 모의실험 알고리즘 및 모델

모의실험을 위한 소프트웨어로 Matlab을 사용하였으며, OSEK-OS와 CAN의 실시간 알고리즘과 모델은 다음과 같다.

2.1 OSEK-OS 모델

OSEK-OS의 스케쥴러(scheduler) 및 각종 서비스를 포함하는 커널을 모델링하고 이벤트 방식(event-driven)의 시뮬레이션 알고리즘을 개발하였다.5)

2.1.1 커널 모의실험 알고리즘

Fig. 1은 커널의 스케줄링 모의실험 알고리즘을 나타낸다. 먼저 시뮬레이션 샘플 시간마다 스케줄링이 일어나 현재 시간에 실행되어야 할 태스크를 결정한다. 실행되어야 할 태스크가 결정되면 해당 태스크가 실행된다. 만약 태스크의 실행결과가 현재 시간에서 다시 스케줄링이 필요한 경우에는 다시 스케줄링 단계로 돌아간다. 그렇지 않으면 다음 단계로 넘어가 태스크 실행 결과로부터 다음 스케줄링이 일어나야 할 시간을 결정한 후 해당 시간이 되면 스케줄링이 일어난다.

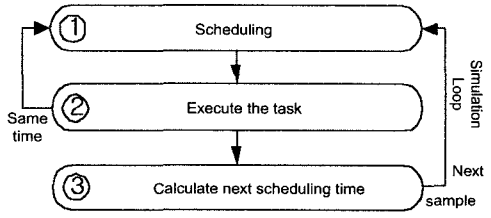

Fig. 1 Simulation algorithm

OSEK-OS는인터럽트를 제외하고는 스케줄링이 일어나야 할 시간의 예측이 가능하므로 위의 방법으로 연계 모의실험이 가능하다. 알고리즘은 Matlab의 s-function을 이용하여 구현하였다.

2.1.2 태스크 모델

분산형 실시간 제어시스템에서 시간 불확실성은 태스크의 실행시간(execution time) 및 우선순위에 변화를 줄 수 있는 OSEK-OS 서비스의 호출에 의하여 발생된다. 연계 모의실험에서 시간 불확실성의 해석을 위하여 한 태스크를 여러 코드 부분으로 나누어 각 코드 부분에실행시간을 부여하였다. 태스크의 총 실행시간은 각 코드부분의 실행시간의 합과 같다. 태스크 모델은 Fig. 2와 같다.

Table 1은 태스크를 코드 부분으로 나눈 예제이다. 예제 코드 1에서는 모든 서비스가 한번에 실행되고 0.03초의 실행시간을 가지는 경우이고, 예제코드 2의 경우에는 각 부분이 0.01초의 실행시간을 가지는 경우이다.

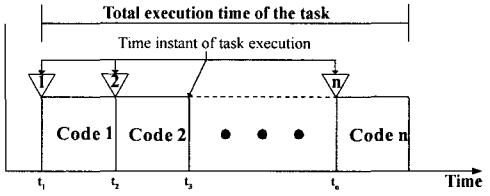

Fig. 2 Task execution model

Table 1 Code example

Example code 1	Example code 2
Switch flag Case 1, SetAbsAlarm(AL,1,1); SendMessage(MsgA); SetEvent(Event); Executime = 0.03; End	Switch flag Case 1, SetAbsAlarm(AL,1,1); Executime = 0.01; End Case 2, SendMessage(MsgA) Executime = 0.01 End Case 3, SetEvent(Event) Executime = 0.01 End

2.2 CAN 프로토콜 모델

실제 시스템에서 CAN 메시지가 전달되는 부분

을 간략히 나타낸 것이 Fig. 3이다. Node A가 Node B로 메시지를 보내는 경우 Node A가 송신 노드가 되고 Node B가 수신 노드가 된다.

실제 CAN 메시지는 Fig. 3에서와 같이 각 노드 내에 있는 CAN 디바이스 드라이버(device driver)의 버퍼를 통하여 전달된다. 이 연구에서는 Fig. 4 같이 CAN 디바이스 드라이버를 CAN 커널 안에 포함하고 있도록 모델링 하였다.

실제 CAN에서 CAN 디바이스 드라이버의 버퍼를 메시지 각각의 고유 버퍼로 가정하고 이 버퍼들을 모두 CAN 커널 안으로 가져와 실제 CAN의 동작을 CAN 커널 안에서 모두 수행하도록 하였다. CAN 커널에서 메시지의 전달 과정은 다음과 같다.

1) 송신 CPU의 태스크에서 CAN 메시지를 보내면 CAN 커널은 메시지를 전송(Tx) 버퍼에 저장한다.

2) CAN 커널의 스케줄러가 전송 버퍼에 있는 메시지들을 우선순위에 따라 전송한다.

3) 전송된 메시지는 CAN 커널에 의하여 정해진 시간만큼 지연된 후 수신(Rx) 버퍼로 저장되어 전송이 완료된다.

CAN 커널의 메시지 지연 모델은 태스크 모델과 비슷하나 메시지 전송 완료 시점은 태스크 모델과 반대로 실행 시간 지연이 끝나는 시간이 된다.

메시지의 전송 버퍼와 수신 버퍼는 큐를 사용하지 않는 경우(Unqueued Message)에는 각 메시지에 버퍼를 하나만 가지고 큐를 사용하는 경우(Queued Message)에는 전송에 필요한 만큼의 버퍼를 가진다.

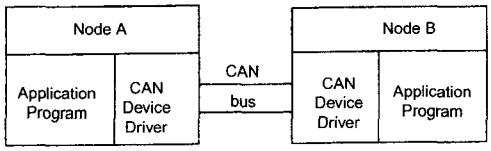

Fig. 3 Communication model

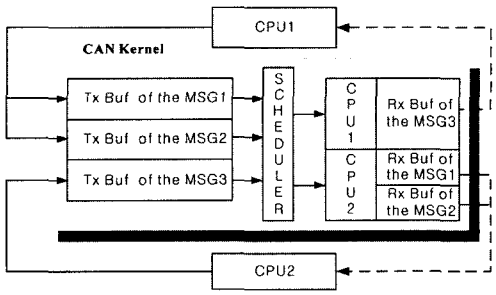

Fig. 4 CAN kernel model

3. 연계 모의실험을 위한 Toolbox

앞에서 제시된 알고리즘을 이용하여 Toolbox를 개발하였다. Toolbox는 Fig. 5에서와 같이 OSEK-OS 커널 블록, CAN 커널 블록 그리고 OSEK-OS 설정을 위한 Object 블록으로 구성되어 있다.

OSEK-OS 커널 블록은 실제 하드웨어에서의 제어기 블록에 해당하며 제어 출력 외에 시간 해석을 위한 태스크 스케줄링 결과를 출력한다. CAN 커널 블록은 CAN 메시지 전송을 담당하고 CAN 메시지의 전송 상태를 출력한다.

OSEK-OS Objects블록들은 OSEK-OS설정을 쉽게 하기 위하여 GUI형태로 개발하였다. 모든 블록들은 Matlab의 마스크를 사용한 서브 시스템(masked subsystem)으로 제작되었으며 커널 블록들은 시스템 함수(s-function)블록으로 제작되었다.

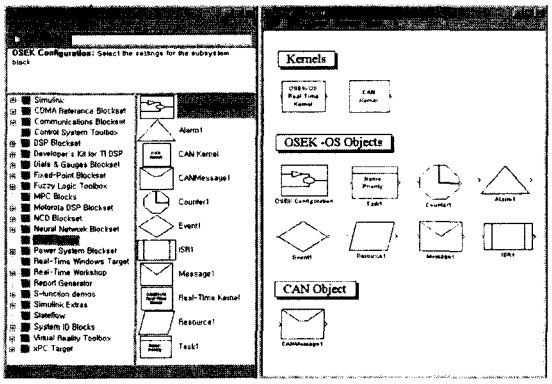

Fig. 5 Toolbox for OSEK-OS and CAN

4. 차체 네트워크 모의실험

내장형 실시간 분산 제어시스템에서의 연계 모의실험을 위하여 자동차의 차체 네트워크 모델을 이용하였다.[6]

4.1 차체 네트워크 모델

차체 네트워크 시스템은 자동차의 Door, Side

mirror, Window 시스템으로 구성되어 있다. 각각의 위치는 DF(Drive Front), PF(Passenger Front), DR(Drive Rear) 그리고 PR(Passenger Rear)로 각 위치마다 OSEK-OS 기반의 컨트롤러가 제어를 하고 각각의 노드는 CAN통신으로 연결되어 있다. DF노드를 제외한 노드들은 자체 입력과 운전자 측에서 전송되는 CAN 메시지를 입력 신호로 가진다.[6]

Fig. 6은 차체 네트워크 시스템의 연계모의실험 모델을 나타낸다. 각 노드들은 OSEK-OS 커널을 가진 CPU로 구성되며 CPU 내부는 Fig. 7과 같이 OSEK-OS Real-time 커널과 OSEK-OS 설정 블록으로 구성되어 있다.

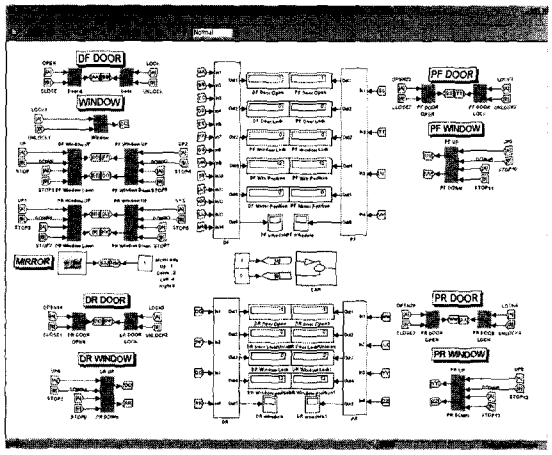

Fig. 6 Body network model

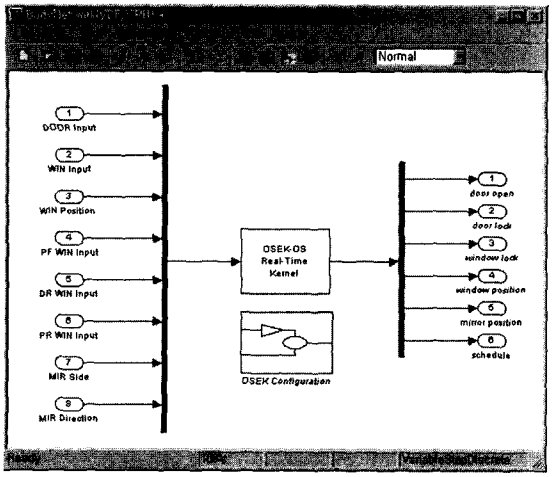

Fig. 7 Inside CPU block

OSEK-OS 설정 블록 안에는 OSEK-OS 설정을 위한 Object 블록들로 구성되어 있다. 이때 블록들을 이용해 Fig. 8과 같이 소프트웨어 구조를 표현할 수도 있다. Fig. 8은 4 노드 중 DF 노드의 OSEK-OS 설정 및 소프트웨어 설계를 보여주고 있다.

DF 노드는 전체 CAN 메시지의 송신자가 되고 나머지DR, PF, PR 노드들은 수신자가 된다. Table 2는 태스크들의 우선순위(Priority)와 역할을 보여주고 있다.

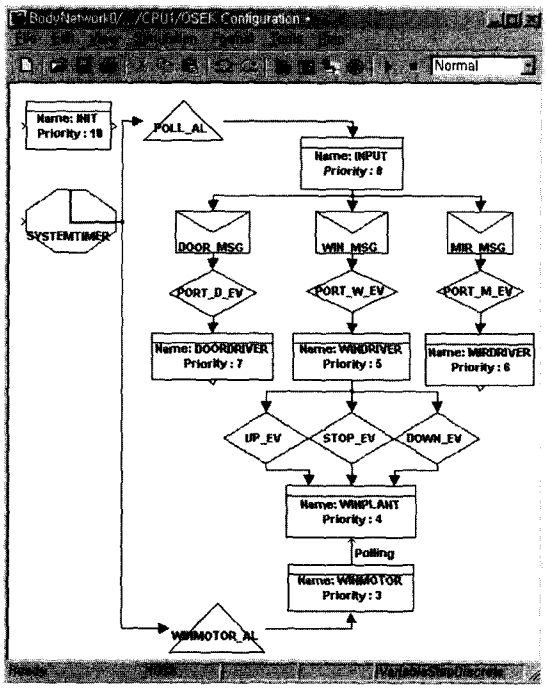

Fig. 8 DF node OSEK-OS configuration

Table 2 Task property

Task	Pri.	역 할
Input	8	DF 노드 : Window, Mirror, Door 입력 체크 후 해당 입력 태스크로 메시지 전송 또는 해당 노드로 CAN 메시지 전송 PF, DR, PR 노드: 자체 입력과 DF노드에서의 CAN 신호 체크 후 해당 태스크로 메시지 전송
Door driver	7	Door driver
Win driver	6	Window driver
Mir driver	5	Mirror driver (DF, PF 노드 Only)

각 노드의 INPUT 태스크는 3ms 의 주기마다 활성화되어 입력을 확인하고 입력신호의 상태 변화에 따라서 WINDRIVER, DOORDRIVER, MIRDRIVER 태스크를 활성화 시켜준다. INIT, WINPLANT, WINMOTOR 태스크는 초기화와 차체 네트워크 시스템의 플랜트 모델링을 위한 태스크들이다.

각 태스크의 실행시간은 하드웨어로 구성된 실제 시스템에서 OSEK-OS의 서비스 중심으로 하드웨어에서 측정한 실행시간을 사용하였다.

모의실험에서는 0.01초에 DF 입력으로 window lock 입력을 주었을 때 CAN 메시지의 데이터 전송속도 변화에 따른 태스크들의 영향을 실험하였다. DF 에서 window lock 입력은 동시에 다른 3 노드에 PF_W_MSG., DR_W_MSG, PR_W_MSG의 CAN 메시지를 보내게 된다. 입력에서부터 태스크들의 연결 관계는 Fig. 9와 같다. 각 메시지의 ID와 전송속도는 Table 3과 같다.

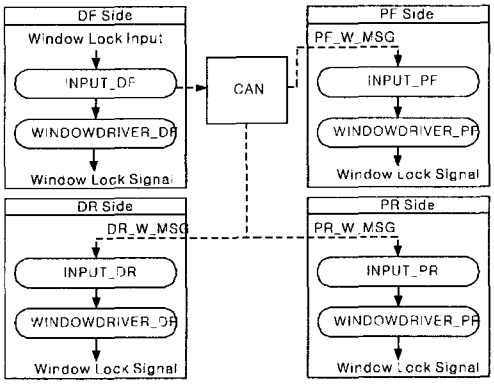

Fig. 9 Data flow

Table 3 CAN message configuration

Message Name	ID	Data length	Baud rate(bit/sec)	
			Case1	Case2
CAN_DL_MSG	1	8 byte	125 K	1 M
CAN_M_MSG	2			
PF_W_MSG	3			
DR_W_MSG	4			
PR_W_MSG	5			

4.2 모의실험 및 해석

모의실험에서는 모의실험결과와 시간 불확실성 해석을 위한 태스크와 CAN 메시지의 스케줄링 결과를 출력한다.

Fig 10, 11은 두 경우의 모의실험 후 얻은 스케줄링 그래프이다. 각 그래프는 DF 노드의 INPUT, WINDRIVER 태스크와 CAN 메시지, 그리고 CAN 메시지에 의해 응답 시간이 변화하는 PF, DR, PR 노드의 INPUT, WINDRIVER 태스크들의 상태 나타내고 있다.

모든 그래프에서 기준값이 높을수록 우선순위가 (Priority) 높다. 태스크 스케줄링 그래프의 경우 해당 기준 값이 suspend state를 나타내고, 여기서 0.25만큼 올라가면 ready state, 0.5는 waiting state, 0.75는 running state를 나타낸다. CAN 메시지의 경우는 각 기준에서 0.2 증가한 상태는 높은 우선순위의 다른 메시지에 의하여 지연되고 있는 상태를 나타내며, 0.5만큼 올라간 상태는 메시지가 전달되는 상태를 나타낸다.

그림에서 모든 INPUT 태스크는 3ms 마다 알람에 의해 실행되고 WINDRIVER 태스크는 waiting 상태에서 이벤트를 기다리다가 INPUT 태스크에서 보낸 메시지에 의해 활성화되어 ready 상태 후 INPUT 태스크가 실행을 마치면 실행된다. INPUT 태스크들이 입력을 받아 실행될 경우에는 실행시간이 길고 입력이 없을 때는 실행시간이 짧은 것은 태스크의 실행시간이 모의실험에서잘 반영되고 있음을 보여준다.

Fig. 10은 DF 노드가 보낸 3개의 CAN 메시지 중에서 PR 노드로 전송되는 CAN 메시지가 다른 메시지들에 의해 메시지 전달이 지연되고 있음을 보여준다. 이로 인해 PR 노드는 0.016초에 CAN 메시지를 확인하고 실행되어 PR 노드의 WINDRIVER 태스크를 활성화 시켜주고 있다.

Fig. 11은 CAN 메시지의 메시지 전송 속도를 125Kbit/sec에서 1Mbit/sec로 바꾸어 테스트한 결과이다. CAN 메시지의 전달 속도 증가로 인해 PR_W_MSG가 0.013초안에 전송이 되어 PR 노드의 INPUT 태스크가 0.013초에 메시지를 받아 실행된다.

실험 결과에서 보듯 플랜트와 제어기의 연계 모의실험에서 태스크의 상태와 CAN 메시지의 상태를 확인 할 수 있다.

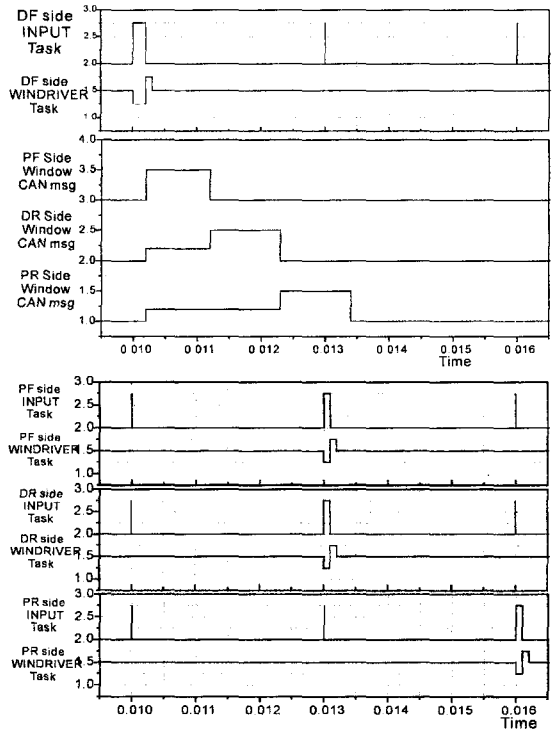

Fig. 10 Scheduling of case 1

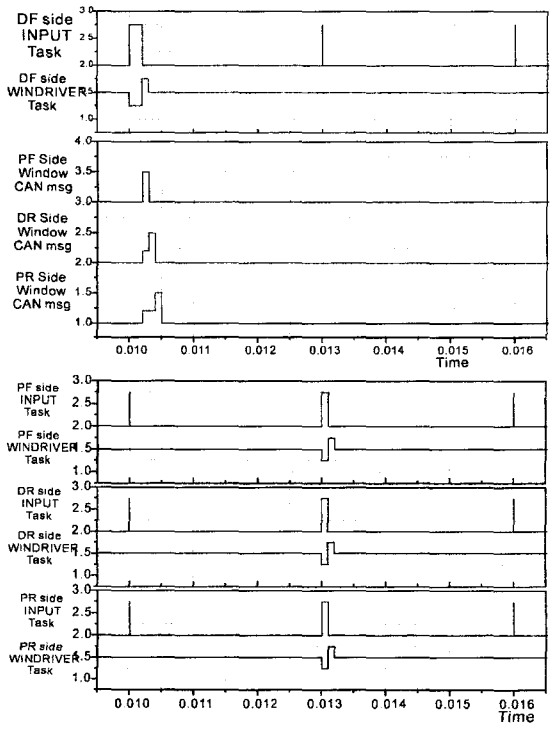

Fig. 11 Scheduling of case 2

5. 결 론

이 연구를 통하여 얻은 주요 결과는 다음과 같다.

1) OSEK-OS와 CAN 프로토콜로 구성된 분산 제어시스템의 실시간 문제에 대한 해석을 수행할 수 있는 모의실험 알고리즘을 개발하였다.

2) 플랜트 모델과의 연계 모의실험을 효율적으로 수행하기 위하여 Matlab에서 사용할 수 있는 도구상자(Toolbox)로 이 알고리즘을 구현하였으며 GUI 환경을 제공하여 사용의 편의를 도모하였다.

3) 실시간 분산 제어시스템의 적용의 예로 자동차 차체 네트워크 시스템을 모델링하여 태스크 및 네트워크 메시지의 스케줄링에 따른 제어 시스템의 시간 문제를 해석하였다.

References

1) B. Wittenmark, J. Nilsson, M. Törngren, "Timing Problems in Real-time Control Systmes," In Proceedings of the American Control Conference, 1995.

2) J. Eker, A. Cervin, "AMatlab Toolboxfor Real-time and Control Systems Co-design," In Proceedings of 6th International Conference on Real-Time Computing Systems and Applications, pp.320-327, 1999.

3) Motorola, OSEK/VDX Operating System Ver2.1, 2000.

4) L. Wolfhard, "CAN System Engineering from Theory to Practical Application," Springer, 1997.

5) Motorola, HC12 OSEK Operating System User's Manual Rev1.7, 1999.

6) M. S. Shin, W. T. Lee, M. S. Sunwoo, "Holistic Scheduling Analysis of a CAN Based Body Network System," Academic Lecture of Electricity and Electronic, ITS of KSAE, pp.111-116, 2001.