한국자동차공학회논문집 제 10 권 제 4 호, pp.175-180

OSEK/VDX 표준과 CAN 프로토콜을 사용한 차체 네트웍 시스템 개발

Development of a Body Network System with OSEK/VDX Standards and CAN Protocol

신 민 석*, 이 우 택*, 선 우 명 호**, 한 석 영**
Minsuk Shin, Wootaik Lee, Myoungho Sunwoo, Seogyoung Han

ABSTRACT

In order to satisfy the requirements of time reduction and cost saving for development of electronic control systems(ECU) in automotive industry, the applications of a standardized real-time operating system(RTOS) and a communication protocol to ECUs are increased. In this study, a body control module(BCM) that employs OSEK/VDX(open system and corresponding interfaces for automotive electronics/vehicle distributed executive) OS for the RTOS and a controller area network(CAN) for the communication protocol is designed, and the performances of the system are evaluated. The BCM controls doors, mirrors, and windows of the vehicle through the in-vehicle network. To identify all the transmitted and received control messages, a PC connected with the CAN communication protocol behaves as a CAN bus emulator. The control system based upon in-vehicle network improves the system stability and reduces the number of wiring harness. Furthermore it is easy to maintain and simple to add new features because the system is designed based on the standards of RTOS and communication protocol.

주요기술용어 : Distributed real-time control system(실시간 분산제어 시스템), Real-time OS(실시간 운영체제), Standardized network(표준 네트웍), CAN communication protocol(CAN 통신 프로토콜)

1. 서 론

자동차 전자제어 기술의 발달과 마이크로 컨트롤러의 성능 향상으로 자동차 내에 사용되는 전자제어 장치들의 수는 갈수록 증가하고 있다. 또한 신차의 개발기간은 점점 짧아지고 차량의 개발 및 생산에 드는 비용 절감과 제어성능의 향상 등에 대한 요구 또한 커지고 있다. 이러한 요구를 만족시키기 위하여 각 전자제어 장치들은 기능에 있어서의 모듈화가 이루어지고 있으며, 효과적이고 체계적인 개발 프로세스와 최적의 시스템 구성을 필요로 한다.[1] 오늘날 자동차 내에서 실시간 분산제어 시스템의 사용이 일반화되고 시스템의 유지, 보수 및 재사용에 대한 필요성이 대두되고 있는 것은 이러한 이유에 기인한

* 회원, 한양대학교 대학원
** 회원, 한양대학교 자동차공학과

것이다. 또한 분산제어 시스템의 효과적인 제어를 위하여 네트웍 기반의 제어시스템 설계기술 연구에 대한 필요성도 커지고 있다. 네트웍을 구성할 때 서로 다른 인터페이스와 프로토콜을 사용하는 분산제어 시스템들의 경우 컨트롤러 간에 발생하는 통신상의 문제를 해결하기 위하여 시스템 표준화에 대한 요구 역시 대두되고 있다. 이러한 요구를 만족시키기 위하여 Bosch는 ISO 표준안으로 지정된 CAN 프로토콜을 제시하였고 유럽의 여러 자동차 관련 업체들은 OSEK/VDX 표준을 제안하였으며 현재 자동차 산업에서 표준으로 자리 잡아가고 있다.[1)]

이번 연구에서는 OSEK/VDX 표준인 OSEK OS(operating system), COM (communication) 그리고 NM(network management)와 실시간 분산제어 시스템의 통신에 있어서 가장 일반적인 프로토콜의 하나인 CAN을 적용한 차체 네트웍 시스템을 구성하여 시스템의 성능을 검증하였다.

2. OSEK/VDX standard & CAN protocol

2.1 OSEK/VDX

OSEK/VDX는 BMW, DaimlerChrysler, Volvo, Volkswagen, Siemens, Bosch 그리고 Motorola 등 유럽의 자동차 관련 업체들을 중심으로 추진되고 있는 자동차 산업의 공동 프로젝트로서 차량의 전자제어 시스템(ECU)을 위하여 개발된 개방형 구조(open-ended architecture)에 대한 업계 표준이다.

OSEK/VDX는 실시간 운영체제(OS), 통신을 위한 상위계층 프로토콜(higher level protocol, COM), 네트웍 관리(NM), 그리고 OSEK/VDX 표준을 구현하기 위한 언어(OSEK Implementation Language : OIL)를 표준으로 정하고 있다.

2.1.1 OSEK/VDX operating system (OS)

OSEK/VDX OS는 컨트롤러가 실시간으로 동작할 수 있는 환경을 제공하며 사용 목적에 따라 네 가지의 컨퍼먼스 클래스(conformance class)로 구분된다. 컨트롤러 내에서 처리되는 프로그램의 기본 단위인 태스크들을 우선 순위에 따라 관리하며 태스크간의 동기화를 위하여 자원 및 이벤트 관리(resource and event management)에 대한 서비스를 제공한다. 이밖에 인터럽트처리(interrupt handling)와, 경고(alarm), 카운터(counter) 그리고 오류처리(error handling) 등을 위한 서비스를 제공한다.[2)]

2.1.2 OSEK/VDX communication (COM)

OSEK/VDX COM은 차량 네트웍에 필요한 상위계층에서의 표준 인터페이스와 프로토콜을 제공한다. 차량내의 통신은 네트웍을 구성하고 있는 ECU간의 통신과 ECU내의 태스크 간의 통신으로 구분되고 OSEK/VDX COM에서는 두 가지의 경우에 대하여 동일한 응용프로그램 인터페이스(Application Program Interface : API)를 제공한다. OSEK/VDX COM은 애플리케이션층(application layer)과, OSEK/VDX NM 그리고 하드웨어의 통신 버스와 연결된다. OSEK/VDX COM은 계층 구조를 기초로 하여 구성되어 있으며 메시지의 전송을 위한 API를 제공하는 상호작용층(interaction layer), 분할 데이터(segmented data)의 전송을 위한 서비스를 제공하는 네트웍층(network layer)과 네트웍에서 사용하는 각 데이터의 전송을 위한 서비스와 네트웍 관리를 위한 서비스를 제공하는 데이터 링크층(data link layer)이 여기에 포함된다.[3)]

2.1.3 OSEK/VDX Network Management(NM)

OSEK/VDX NM은 네트웍 시스템에서 통신이 올바르게 일어나고 있는가를 감시하고 관리하는 역할을 한다. 네트웍을 관리하는 방식에는 직접 네트웍 관리(Direct Network Management) 및 간접 네트웍 관리(Indirect Network Management) 두 가지 방식이 있다. 직접 네트웍 관리 방식은 네트웍을 형성하고 있는 모든 노드들이 다른 노

드에 의해 모니터링 되는 방식이다. 이를 위하여 노드들은 논리 링(logical ring)을 구성하고 논리 링간에 별도의 네트웍 메시지(NM message)를 사용한다. 간접 네트웍 관리 방식은 네트웍 메시지를 사용하는 것이 아니라 주기적으로 전송되는 애플리케이션 메시지를 이용한다.[4]

2.2 CAN protocol

CAN은 실시간 분산제어 시스템에서 데이터를 전달하기 위한 직렬 통신 프로토콜로서 ISO 11898(for high-speed application) 및 ISO 115192 (for lower-speed application) 표준이다.

CAN 프로토콜은 CSMA/CD+AMP(Carrier Sense Multiple Access with Collision Detection and Arbitration on Message Priority)방식으로 메시지를 전달하므로 동시에 네트웍을 구성하고 있는 모든 노드에서 수신이 가능하다. CAN 메시지의 전달은 두 신호 라인의 전압차를 이용하여 이루어지기 때문에 노이즈에 강한 내성을 가진다. 또한 각 메시지의 CRC 비트를 이용하여 오류 검출 능력이 우수하고, 최대 1Mbps의 빠른 전송 속도를 갖고 있기 때문에 자동차에의 적용이 점점 늘어가고 있다. CAN에서는 메시지 프레임을 데이터 전달을 위한 데이터 프레임(data frame)과 데이터 요청을 위한 원격 프레임(remote frame), 오류 발생을 알리기 위한 에러 프레임(error frame), 그리고 CAN 컨트롤러가 연속된 프레임 사이에서 이전에 전송된 메시지의 처리를 마치지 못하였을 경우 메시지의 재전송을 요청하고 다음 메시지의 전송을 지연시키기 위한 과부하 프레임(overload frame)으로 구분한다. CAN 메시지의 형식은 중재 영역(arbitration field)의 크기에 따라 표준형과 확장형의 두 가지로 나뉘며, 각각 11비트와 29비트의 중재 영역을 갖는다. 중재 영역은 각 메시지의 식별자(identifier)로 사용되고 식별자는 전체 네트웍에서 고유하게 할당된 것으로 크기가 작을수록 높은 우선 순위를 갖는다. 하나의 CAN 메시지에서 전송 가능한 데이터 영역의 크기는 최대 8바이트이다.[5,6]

3. 차체 네트웍 시스템 개요

모토로라 OSEK[7,8]을 이용하여 OSEK/VDX 표준에 따른 차체 네트웍 시스템을 구성하였다. 여기서 구현된 차체 네트웍 시스템은 자동차문의 개폐와 잠금 및 해제 그리고 차창의 위치를 각 문에서 조정할 수 있다. 또한 이 시스템은 운전자에 의하여 모든 문의 잠금 및 해제, 차창의 위치조정과 잠금 및 해제, 그리고 두개의 후사경의 위치를 원격 조정할 수 있도록 설계된 시스템이다. 시스템의 하드웨어(hardware) 구성과 노드간의 CAN 버스를 이용한 네트웍 구성은 Fig. 1과 같다. 전체 시스템은 자동차문을 중심으로 네 개의 노드들로 이루어져 있으며 각 노드는 입력신호를 발생시키는 부분과 이를 처리하여 출력으로 내보내는 컨트롤러로 구성된다.

컨트롤러들은 CAN 버스로 연결되어 있으며 이를 통해 노드간의 통신이 이루어진다. 시스템을 PC와 연결하여 에뮬레이션(emulation)이 가능하도록 하기 위하여 가상 노드를 추가하였다. 이 노드 역시 CAN 버스로 연결되어 다른 노드와 통신이 가능하다. 가상 노드의 역할을 하는 PC는 계기판의 역할을 포함하는 노드이다. 실제 자동차문, 차창, 그리고 후사경의 기능을 조작하는데 필요한 I/O들은 토글 스위치와 푸시 버튼 그리고 발광 다이오드를 이용하여 구현하였다. 자동차

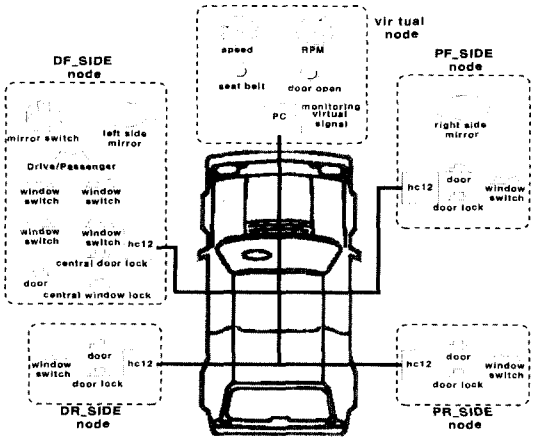

Fig. 1 Body network system

문을 나타내고 있는 네 노드들 간의 관계에서 운전자측 노드(DF_SIDE)는 메시지 송신자의 역할을 하며 나머지 세 노드는 수신자의 역할을 한다. 그리고 이들 네 노드는 PC와 연결된 가상 노드에 대하여 자동차문의 개폐에 대한 메시지를 전송하는 역할을 한다.

시스템을 구성하기 위하여 모토로라에서 제공하는 HC12를 위한 OSEK OS와 COM/NM과 68HC912D60 마이크로 컨트롤러를 이용하였고, HIWARE사에서 제공하는 HC12용 C 컴파일러와 링커를 사용하였다. 그리고 PC와의 연결을 위하여 병렬 포트를 이용하여 통신을 하는 EMICROS사의 CANport를 사용하였다.

4. PC에서의 Emulation

PC상에서 에뮬레이션 할 때 제공되는 사용자 인터페이스는 Fig. 2와 같다. 이것은 노드의 구성과 각 노드에서의 스위치 입력을 나타낸다. 이 사용자 인터페이스는 네트웍 시스템과 CANport를 통하여 연결되어 PC상에서 CAN 메시지의 움직임을 확인할 수 있을 뿐만 아니라 CAN 메시지를 발생시켜 네트웍 시스템을 동작시킬 수도 있다. 이 부분에 해당하는 프로그램은 비주얼 베이직을 사용하였다.

5. CAN 메시지 ID 지정

차체 네트웍 시스템에서 필요로 하는 신호(signal)들을 하나의 메시지로 표현한 것이 Table 1

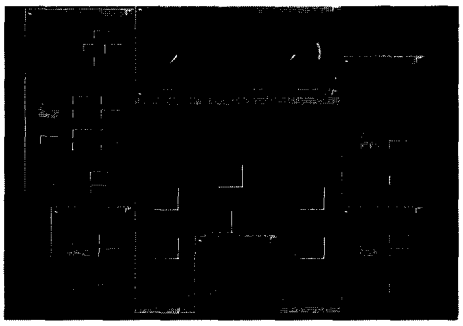

Fig. 2 Graphic user interface at PC

이고 이들 신호를 송신자와 수신자 그리고 각 메시지들이 하는 역할 등을 고려하여 재구성한 것이 Table 2이다. 통신 속도는 125kbps이고 CAN 메시지의 주기는 INPUT_T 태스크가 활성화되는 주기인 20ms로 하였다. 메시지의 대기시간과 실제 전송되는데 걸리는 시간의 합을 의미하는 응답시간은 메시지의 오버헤드(over head)를 고려한 최악의 경우(worst case)에 대한 것으로 Ken Tindell에 의하여 제안된 방법으로 해석한 결과이다.[9] 두 경우 모두 스케줄링이 가능하며 버스 사용률은 비트 단위로 메시지를 정하였을 경우

Table 1 List of required CAN messages (T : period, R : response time)

No	Signal Description	Size /bits	R / ms	T /ms	ID	From	To
1	Door Lock	1	1.04	20	0x1	DF_SIDE	Others
2	Window Lock	1	1.56	20	0x2	DF_SIDE	Others
3	DF Door Close	1	2.08	20	0x10	DF_SIDE	.PC
4	DF Door Open	1	2.6	20	0x11	DF_SIDE	PC
5	PF Door Close	1	3.12	20	0x20	PF_SIDE	PC
6	PF Door Open	1	3.64	20	0x21	PF_SIDE	PC
7	PF_Win_Stop	1	4.16	20	0x24	DF_SIDE	PF_SIDE
8	PF_Win_Up	1	4.68	20	0x25	DF_SIDE	PF_SIDE
9	PF_Win_Down	1	5.2	20	0x26	DF_SIDE	PF_SIDE
10	Mirror_Up	1	5.72	20	0x28	DF_SIDE	PF_SIDE
11	Mirror_Down	1	6.24	20	0x29	DF_SIDE	PF_SIDE
12	Mirror_Left	1	6.76	20	0x2A	DF_SIDE	PF_SIDE
13	Mirror_Right	1	7.28	20	0x2B	DF_SIDE	PF_SIDE
14	Mirror_Stop	1	7.8	20	0x2C	DF_SIDE	PF_SIDE
15	DR Door Close	1	8.32	20	0x30	DR_SIDE	PC
16	DR Door Open	1	8.84	20	0x31	DR_SIDE	PC
17	DR_Win_Stop	1	9.36	20	0x34	DF_SIDE	DR_SIDE
18	DR_Win_Up	1	9.88	20	0x35	DF_SIDE	DR_SIDE
19	DR_Win_Down	1	10.4	20	0x36	DF_SIDE	DR_SIDE
20	PR Door Close	1	10.92	20	0x40	PR_SIDE	PC
21	PR Door Open	1	11.440	20	0x41	PR_SIDE	PC
22	PR_Win_Stop	1	11.96	20	0x44	DF_SIDE	PR_SIDE
23	PR_Win_Up	1	12.48	20	0x45	DF_SIDE	PR_SIDE
24	PR_Win_Down	1	12.48	20	0x46	DF_SIDE	PR_SIDE

Table 2 List of packed CAN messages (T : period, R : response time)

No	Message	Size /bits	R /ms	T /ms	ID	From	To
1	DoorLockMsg	1	1.04	20	0x10	DF_SIDE	Others
2,3	DFDoor	1	1.56	20	0x11	DF_SIDE	PC
4,5	PFDoor	1	2.08	20	0x21	PF_SIDE	PC
20~24	MIRMsg	4	2.6	20	0x22	DF_SIDE	PF_SIDE
10,11, 12,13	PFWinMsg	3	3.12	20	0x24	DF_SIDE	PF_SIDE
6,7	DRDoor	1	3.64	20	0x31	DR_SIDE	PC
10,14, 15,16	DRWinMsg	3	4.16	20	0x34	DF_SIDE	DR_SIDE
8,9	PRDoor	1	4.68	20	0x41	PR_SIDE	PC
10,17, 18,19	PRWinMsg	3	4.86	20	0x44	DF_SIDE	PR_SIDE

62.4%, 이를 재구성한 경우 23.4%이다. CAN 메시지 식별자의 결정은 각 노드와 입력을 기준으로 정한다. 메시지 식별자의 결정시 Fig. 1의 DF_SIDE, PF_SIDE, DR_SIDE와 PR_SIDE 노드는 각각 0×10, 0×20, 0×30와 0×40으로 표시하고, 자동차문, 후사경 그리고 차창에 관련된 입력은 각각 0×01, 0×02와 0×04로 표시한다. 즉, PF_SIDE의 후사경에 관련된 메시지의 경우 식별자는 0×되고 DR_SIDE의 차창에 관련된 메시지의 경우 식별자는 0×34가 되는 식이다.

6. 태스크 구현

Fig. 3은 각 노드에서 태스크의 구현을 보여준다. 태스크들은 그 기능에 따라 크게 세 가지로 구분된다. 먼저 컨트롤러와 출력을 초기화하는 초기화 태스크(INIT_T)와 주기적으로 입력을 점검하는 입력 태스크(INPUT_T), 그리고 입력에 따라 해당하는 출력을 내보는 드라이버 태스크(DOORDRIVER_T, WINDRIVER_T, & MIRDRIVR _T)로 나뉜다. 또한 노드의 입력은 각 노드의 스위치에 의한 입력과 CAN 메시지에 의한 입력 두 가지로 나뉜다. INPUT_T 태스크는 수신 노드에서는 두 입력을 구분하여 동작하지만 송신 노드에서는 노드의 스위치에 의한 입력만을 가지고 다른 노드를 동작 시키기 위한 메시지를 보내는 역할을 한다.

INIT_T 태스크는 각 노드의 상태를 초기화하고 INPUT_T 태스크는 주기적으로 입력을 점검하여 해당 드라이버 태스크에 메시지를 보낸다. 또 각 드라이버 태스크들은 이 메시지들이 발생시키는 이벤트에 의해 활성화된다. 드라이버 태스크들은 현재 상태와 입력에 따라 출력 신호를 결정한다. 특히 차창 신호의 경우 막대 발광 다이오드를 이용하여 차창의 현재 위치를 에뮬레이션 하기 위하여 WINDOW_T 태스크와 LED_T 태스크를 추가하였다.

아래의 Table 3은 INPUT_T 태스크에서 처리되는 메시지를 정리한 것이다.

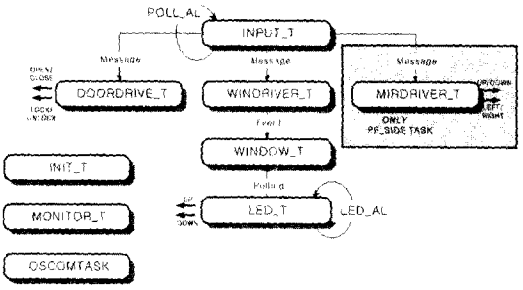

Fig. 3 Task diagram

Table 3 The list of managed messages in INPUT_T (L : Local, N : Network message)

Message	Type	Sender	Receiver	ID
DOOR_MSG	L	INPUT_T	DOORDRIVER_T	
WIN_MSG	L	INPUT_T	WINDRIVER_T	
MIR_MSG	L	INPUT_T	MIRDRIVER_T	
DL_MSG	N	DF_SIDE	PF, DR, PR_SIDE	0x10
MIR_MSG	N	DF_SIDE	PF_SIDE	0x22
DF_D_MSG	N	DF_SIDE	PC	0x11
DR_D_MSG	N	DR_SIDE	PC	0x31
DR_W_MSG	N	DF_SIDE	DR_SIDE	0x34
PF_D_MSG	N	PF_SIDE	PC	0x21
PF_W_MSG	N	DF_SIDE	PF_SIDE	0x24
PR_D_MSG	N	PR_SIDE	PC	0x41
PR_W_MSG	N	DF_SIDE	PR_SIDE	0x44

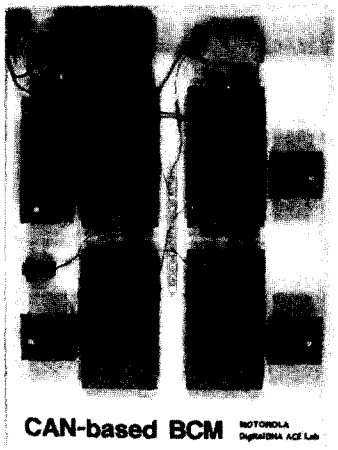

Photo. 1 A proto body network system

전체 시스템은 OSEK COM에 의해 CAN 메시지를 이용하여 통신이 이루어지고, NM에 의해서 네트웍이 모니터링 된다. 이 시스템에서 사용되는 모든 메시지는 이벤트에 의해 발생하는 메시지이므로 시스템을 모니터링하기 위하여 논리 링을 이용하는 직접 네트웍 관리 방식을 적용하였다. COM과 NM에 해당하는 부분은 OSCOMTASK에 의하여 처리된다.

MONITOR_T는 NM에 의하여 네트웍을 이루고 있는 노드의 구성이 바뀌었을 때 활성화되어 현재 네트웍의 상태를 알려주는 태스크이다.[3,4)]

실제 제작된 차체 제어 시스템은 Photo. 1과 같다.

7. 결론 및 향후 연구 계획

이 연구에서는 자동차 업계의 표준 실시간 운영체제와 통신 프로토콜을 사용한 네트웍 기반의 차체 네트웍 제어 시스템을 설계 및 구현하였다. 설계된 차체 네트웍 시스템은 기존의 시스템에서 와이어 하니스(wire harness)로 연결되어 있던 부분들을 네트웍으로 대치함으로써 와이어 하니스 수를 줄일 수 있을 뿐만 아니라 시스템의 통신 메시지를 항시 모니터링 할 수 있어 시스템의 안정성도 높일 수 있다. 또한 표준화된 운영체제와 통신 프로토콜을 사용함으로써 시스템의 유지, 보수 및 기능의 확장이 용이하다. 이번 연구를 통해 CAN을 사용한 네트웍 시스템의 구축 시 PC기반의 에뮬레이션 노드를 사용하여 효과적으로 이용할 수 있는 방법을 제시하고, 사용하였다.

참 고 문 헌

1) OSEK/VDX Binding specification V1.2.
2) OSEK/VDX Operating System V2.1r1.
3) OSEK/VDX Communication V2.2.2.
4) OSEK/VDX Network Management V2.51.
5) Wolfhard Lawrenz, CAN System Engineering from Theory to Practical Application, Springer, 1997.
6) Boshc Controller Area Network(CAN) Version 2.0.
7) Motorola, HC12 OSEK Operating System User's Manual V.1.7.
8) Motorola, OSEK COM/NM User's Manual V 1.0.
9) K. Tindell "Fixed Priority Scheduling of Hard Real-time Systems," PhD Thesis, Department of Computer Science, University of York, 1994.
10) B. D. Emaus, Current Vehicle Network Architecture Trend 2000 SAE 2000-01-0146, 2000.
11) L. Johansson, P. Broqvist, J. Ohlsson Efficient CAN Based Automotive Control Systems 2000-04-11, 2000.