논문 2013-08-20

AUTOSAR CAN Interface 확장을 이용한 ECU의 소프트웨어 업데이트 방법

(Fast ECU Software Update Algorithm using by Extending AUTOSAR CAN Interface)

김 종 욱, 백 장 운, 권 기 구, 이 석 규*
(Jong-Uk Kim, Jang-Woon Baek, Kee-Koo Kwon, and Suk-Gyu Lee)

Abstract : This paper proposes a novel ECU (Electronic Control Units) update algorithm for AUTOSAR based automotive embedded system. The proposed algorithm provides fast and easy ECU update by extending AUTOSAR CAN Interface. The proposed system removes the update sequences from PDUR to RTE (ECU update program), and it stops other ISRs and operating system in order to reduce unnecessary context switching time. In experimental results, we can see that the proposed algorithm reduces update time.

Keywords : AUTOSAR, CAN If, ECU Update algorithm, UDS

I. 서 론

운전자의 편의성 및 안전성에 대한 요구 사항이 많아짐에 따라 자동차에 ECU(Electronic Control Units)의 사용이 급격하게 증가하고 있다. 기존 차량은 엔진, 브레이크, 파워트레인과 오디오 정도에만 ECU가 사용되었으나 현재 최고급사양의 차량에는 80여개의 ECU가 사용되고 있다. 많은 ECU가 사용되고 다양하고 복잡한 기능을 요구하는 시스템이 증가함[1]에 따라 예전처럼 차량에 사용되는 SW의 유지 보수와 호환성의 공유 및 개발 속도에 대한 연구가 시도되었으며[2], 그 결과의 하나로 AUTOSAR(AUTomotive Open System ARchitecture)라는 차량전장용 소프트웨어표준플랫폼의 적용이 의무화 되고 있다[3, 4].

AUTOSAR 플랫폼 사용이 의무화 되면서 차량 ECU에 탑재되는 플랫폼의 운영체제 업데이트, SW 기능 개선, 오류 정정 및 보안 패치 등의 ECU 업데이트가 중요한 문제가 되고 있다. 많은 ECU가 장착이 되다 보니 개별적으로 분리하여 ECU를 업데이트하는 것은 비용과 시간이 많이 소요된다. 그래서 현재는 일반적으로 차량에는 CAN(Controller Area Network), LIN(Local Interconnect Network), 및 FlexRay 통신 등 ECU간의 공유된 네트워크를[5] 이용하여 업데이트할 데이터를 전송하고 ECU 업데이트를 수행한다.

현재 국내외의 대부분의 차량에서는 CAN을 이용하여 ECU를 업데이트하고 있다. AUTOSAR 플랫폼에도 CAN 프로토콜을 이용한 ECU 업데이트 관련된 기술이 명시되어 있는데 메모리에 데이터를 저장하는 방법에는 크게 두 가지가 존재한다.

첫 번째는 소프트웨어 컴포넌트라는 응용소프트웨어에서 CAN 메시지를 입력받아 NVRAM Manager를 호출하여 처리하는 방법이고 두 번째는 DCM (Diagnostic Communication Manager)에서 입력된 CAN 메시지의 내용을 NVRAM Manager에 전달하여 메모리에 저장하는 방식이다. AUTOSAR 표준에서는 이 메시지들을 DCM이나 소프트웨어 컴포넌트가 비동기적인 방법으로 RAM에서 필요한 내용을 복사하여 EEPROM이나 Flash에 저장하도록 하고 있다.

AUTOSAR 플랫폼은 많은 표준의 사항들을 요구하고, Layer 층을 이용하여 많은 내부 메시지를 사용하여[6, 7] 많은 RAM의 자원을 사용한다. 하

* Corresponding Author (sglee@ynu.ac.kr)
Received: 15 Feb. 2013, Revised: 18 Mar. 2013, Accepted: 08 Apr. 2013.
J.U. Kim, B.W. Baek, K.K. Kwon : ETRI
S.G. Lee: Yeungnam University

ISSN : 1975-5066

지만 RAM의 크기가 제한되어 있는 상태에서 ECU 업그레이드 데이터 저장을 위해 많은 메모리 버퍼 할당이 필요로 하는 문제가 있다. 또 비동기 방식으로 저장하기 때문에 데이터를 수신하고 메모리에 저장할 때까지의 시간지연이 발생한다. 따라서 업데이트할 데이터가 계속 수신될 경우 이를 저장하기 위해 더 많은 RAM 공간이 필요하기 때문에 대용량의 소프트웨어 업데이트를 하기에는 많은 문제가 있다.

본 논문에서는 이러한 문제를 해결하고자 AUTOSAR CAN If(Interface)모듈의 확장을 통하여 동기화방식으로 메모리에 저장하고, 메모리의 버퍼 또한 최소화 하고자 하였다.

II. 배경

1. AUTOSAR 모듈 소개

AUTOSAR에서는 MCAL (Microcontroller Abstraction Layer)이라 부르는 모듈들과 ECAL(ECU abstraction Layer)이라 부르는 모듈들 간의 유기적인 연결을 통해 ECU 업데이트를 한다. 이 모듈들은 CAN 통신과 관련된 통신 모듈과 메모리 관련 모듈로 나눌 수 있다.

통신 관련된 모듈에는 CAN, CAN IF(CAN Interface), CAN Tp(CAN Transport Layer), PDUR (PDU Router), COM(Communication) 모듈이 있다. 이 모듈들은 내부의 데이터 연결을 위하여 PDU ID를 메시지 별로 만들어 관리한다. CAN 드라이버는 AUTOSAR MCAL 영역에 존재하며 CAN 버스 상에 하드웨어로 연결되어 컨트롤러의 하드웨어 레지스터를 제어를 하며 CAN IF에 데이터 및 ID를 전달하는 기능을 담당한다[8]. CAN IF는 CAN 드라이버에서 전달된 정보를 바탕으로 PDU(Protocol Data Unit) ID 기반으로 상위 계층과 연결하는 기능을 한다. CAN Tp는 입력 및 출력 데이터의 조합 및 분할의 기능을 담당한다. COM은 입력된 데이터의 시그널 분할, 출력될 데이터의 시그널 조합 및 전반적인 통신의 시간과 데이터를 관리하며 RTE(Run-Time Environment) 데이터를 전달하거나 수신하는 역할을 담당한다. PDUR(PDU Router)은 COM, CAN Tp 등으로 정의된 데이터를 전달하는 기능을 한다.

메모리 관련 모듈에는 NvM(NVRAMManager), Fee(Flash EEPROM Emulation), EA(EEPROM Abstraction), Mem If(Memory Abstraction Interface), Fls(Flash), Eep(EEPROM)가 필요하다. 이 모듈들은 메모리를 블록화 하고 각각의 블록 ID를 만들어 내부적으로 관리할 때 사용한다. Eep 드라이버는 EEPROM 메모리의 읽기, 쓰기, 삭제에 대한 관리를 비동기화 방식으로 동작하는 서비스를 제공한다. EA은 EEPROM 메모리에 대한 하드웨어의 추상화를 통한 관리를 한다. Fls 드라이버는 FLASH 메모리의 읽기, 쓰기, 삭제에 대한 관리를 비동기화 방식으로 동작하는 서비스를 제공한다. Fee은 플래시 메모리에 대한 하드웨어의 추상화를 통한 관리를 한다. Mem If는 메모리의 분기 작업 및 EA나 Fee로 데이터를 전달하는 기능을 담당한다. NvM은 메모리 전체를 블록 단위로 관리하여 FLASH 및 ROM을 관리하는 기능을 담당한다.

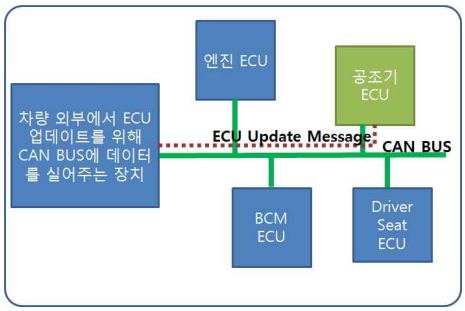

그림 1. CAN BUS상에서 MCU 업데이트 메시지의 흐름
Fig. 1 Flow of the MCU update message on the CAN BUS

그 외에도 RTE (Runtime Environment)와 DCM (Diagnostic Communication Manager) 모듈이 필요하다. RTE는 ECAL에서 전달된 데이터를 소프트웨어 컴포넌트(Software Component)에 전달하는 역할 뿐만 아니라 System 전반적으로 TASK의 동작 및 통신 모듈의 데이터를 상위 계층과 연결하고 관리의 해주는 역할을 담당한다. DCM은 진단 통신 관리 기능과 세션 관리 등의 기능을 하고 진단 관련 응답메시지를 만드는 역할을 담당한다.

2. 소프트웨어 컴포넌트를 이용한 업데이트

그림 1는 CAN BUS상에서 업데이트 하려고 하는 MCU까지의 메시지 흐름을 보여준다. ECU를 업데이트할 데이터는 CAN BUS를 통해 해당 ECU로 전달이 된다. 해당 ECU는 전달된 데이터를 버퍼에 저장하고, 응답 메시지를 만들어 다시 CAN 드라이

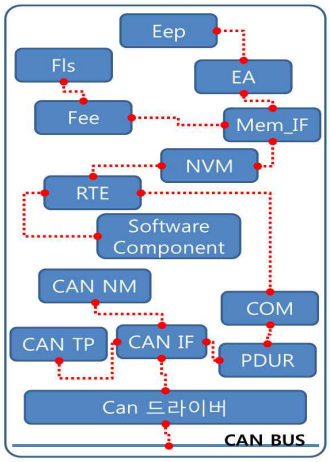

그림 2. Software Component에서 ECU 업데이트 구조
Fig. 2 ECU Update Structure of Software Component

버에 전달하게 된다. CAN 드라이버는 ECU 소프트웨어 업데이트에 대한 응답을 다시 CAN BUS로 보낸다.

그 후 소프트웨어 컴포넌트는 정의된 TASK에서 RTE를 통해 NvM을 호출하고, NvM은 Eep 드라이버까지 호출하여 업데이트 데이터를 EEPROM에 저장한다. 그림 2는 소프트웨어 컴포넌트를 이용한 MCU 업데이트에 필요한 AUTOSAR의 각 모듈간의 관계를 표현한 것이다. AUTOSAR에는 드라이버와 시스템 서비스에서 주기적으로 동작을 하는 TASK들을 묶어 BSW(Basic Software) TASK 한 개에서 제공을 한다. 이 BSW TASK내에 Eep에서 제공하는 메인 함수(Eep_MainFunction())가 실행이 되고 EEPROM에 업데이트 데이터를 저장한다.

그림 3은 시간의 흐름에 따라 소프트웨어 컴포넌트에서 업데이트 하는 방법을 표현한 것이다. CAN 드라이버를 통해 업데이트할 메시지가 입력이 되고, 통신 모듈들을 거쳐 소프트웨어 컴포넌트까지 호출이 된다. 소프트웨어 컴포넌트 모듈은 다시 메모리 모듈을 호출하여 메모리 복사 작업을 예약을 하고 BSW TASK에서 실제 메모리를 저장하는 흐름을 가지고 있다.

3. DCM을 이용한 업데이트

AUTOSAR에서는 DCM 모듈을 이용하여 ECU 업데이트를 할 수 있다. ECU 소프트웨어 업데이트

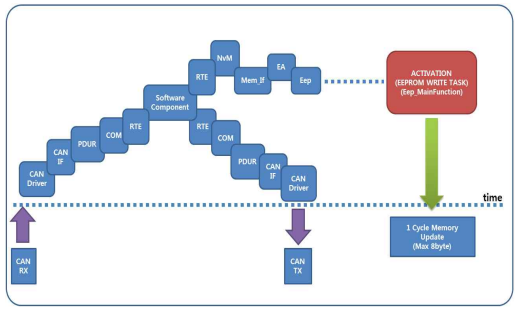

그림 3. Software Component를 이용한 ECU 업데이트의 순서
Fig. 3 ECU Update Procedure Using Software Component

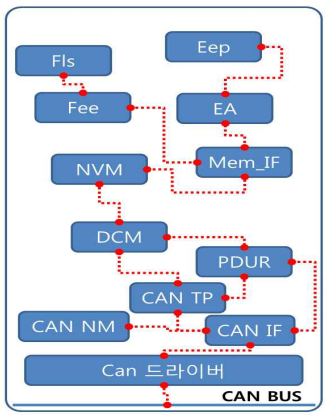

그림 4. AUTOSAR DCM을 이용한 ECU 업데이트의 구조
Fig. 4 ECU Update Structure using AUTOSAR DCM

에 대한 CAN 메시지는 CAN If를 거쳐 CAN Tp에 전달되고, CAN Tp는 버퍼를 만들어 데이터를 저장한다. 저장 후 CAN Tp는 DCM으로 ECU 업데이트 데이터를 전달하고 DCM은 데이터를 분석 후 자동으로 응답 메시지를 만들어 CAN 드라이버에 전달 작업과 동시에 NvM을 호출하여 메모리 Write 작업을 요청한다.

그림 4는 DCM을 이용한 MCU 업데이트에 필요한 AUTOSAR의 각 모듈간의 관계를 표현한 것이다. 기본적인 흐름은 소프트웨어 컴포넌트와 흡사하지만 RTE를 거치지 않고 직접 메모리 모듈에 접근을 하여 업데이트하는 방법이다. DCM을 이용한 ECU 업데이트 방법의 장점은 ISO 14229-1에 적

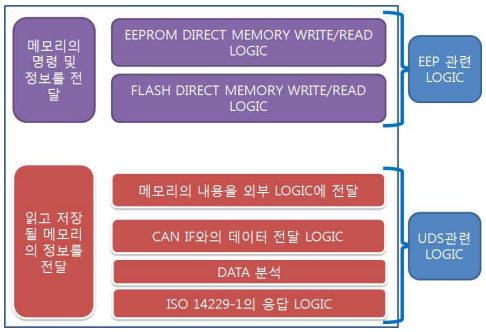

그림 5. EUM 블럭 다이어그램
Fig. 5 Blok diagram of EUM MODULE

합한 CAN 응답을 자동으로 만들어 주고, 업데이트 데이터에 버퍼를 자동으로 관리해주는 장점이 있다. 동적메모리 할당을 이용할 수 없는 AUTOSAR 플랫폼 구조에서 버퍼를 만들 때는 정확한 데이터 크기를 예측 할 수 없고 비동기 방식으로 EEPROM이나 Flash 메모리에 저장하기 때문에 크게 버퍼를 할당시키는 방법 이외에는 존재하지 않는다.

Ⅲ. 제안 알고리즘

소프트웨어 컴포넌트를 이용한 ECU 업데이트 방법이나 DCM을 이용한 업데이트 방법에서 문제점은 메모리 버퍼와 동작시간이다. 많은 시간을 소모하는 것은 Eep 드라이버의 표준 함수인 메인 함수(*Eep_MainFunction()*)의 비동기화로 동작이 되는 부분이다. 대기 시간을 줄이기 위해 주기를 줄이면 실제 동작하지 않더라도 메인 함수가 계속 호출되어 다른 응용에 지속적인 방해를 하거나, 불필요한 자원을 소모하게 되는 결과를 야기한다.

또 통신이나 메모리의 관리를 위해서 통신 모듈에서는 PDU ID를 만들어 메시지 이동을 관리하고 있고, 메모리 관리 모듈에서는 블록 ID를 만들어 관리하고 있다. 따라서 대용량의 업데이트 데이터를 전송하게 되면 이들 ID에 대한 검색 작업과 모듈간의 데이터 교환 작업이 과다하게 발생하여 많은 시간을 소요하게 된다.

앞에서 언급한 문제를 해결하기 위해 본 연구에서는 기존의 AUTOSAR 표준의 문제점을 보안하고 최소의 시간과 자원으로 대용량의 ECU업데이트를 지원하는 ECU 업데이트 알고리즘을 제안한다.

제안하는 알고리즘은 AUTOSAR CAN If 확장을 통해 EUM(ECU Update Module)로 연결하고,

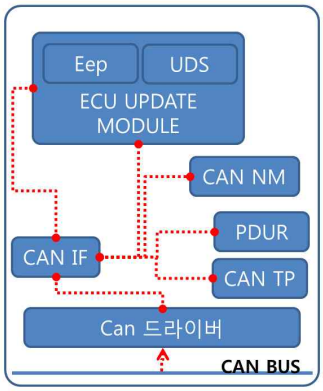

그림 6. ECU 업데이트의 구조
Fig. 6 ECU Update Structure

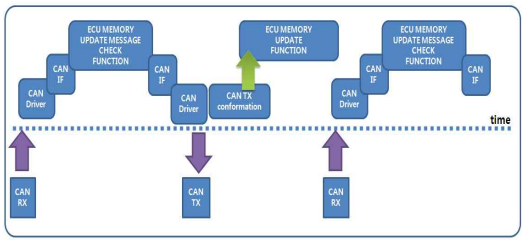

그림 7. 제시한 알고리즘을 이용한 ECU 업데이트의 순서
Fig. 7 ECU Update Procedure of the Proposed Algorithm

빠른 ECU 업데이트를 제공하여 불필요한 검색작업을 줄인다.

ECU 업데이트 모듈(ECU Update Module, EUM)은 내부에서 UDS(Unified Diagnostic Services) 서비스를 지원하는 UDS 모듈과 EEPROM 읽기, 쓰기 오퍼레이션을 수행하는 Eep 모듈을 결합한 모듈이다. 그림 5에는 EUM 내부 모듈의 블록을 표현한 것이다. CAN If에서 입력된 데이터 내부를 확인하고 파싱하는 작업 후 UDS 표준에 맞게 응답메시지를 만들어 CAN If에 전달하고 데이터를 저장한다. 그리고 메모리에 저장 될 데이터를 EEP 관련 로직으로 저장된 데이터의 주소와 길이 정보 및 저장할 메모리의 주소정보를 전달한다. EEP 로직에서는 UDS 로직으로 입력받은 정보를 즉시 메모리에 저장한다. EUM 모듈의 장점은 PDU ID의 검색이나 블록 ID의 검색이 없으며, CAN 메시지의 빠른 응답으로 다음 메시지의 기다림이 적다. 또한, 버퍼의 관리가 기존의 AUTOSAR

표 1. 시뮬레이션 파라미터
Table 1. Simulation Parameters

Program logic and motion	TIME(ms)
Can driver rx processing	0.1
Can If rx processing	0.05
PDUR and COM RX processing	0.15
RTE service	0.05
Can driver tx processing	0.1
Can If tx processing	0.05
PDUR and COM tx processing	0.1
CAN Tx conformation(Standard)	0.3
Software Component processing	0.3
CAN Tx conformation	0.1
NvM processing	0.8
Mem If processing	0.4
EA processing	0.5
Eep processing	0.3
Context switching	0.04
Timer interrupt processing	0.1
CAN TX conformation isr delay	0.1
CAN Transmission delay	2
Period of timer interrupt	1
Eep Main Function Period	10
Eep Main Function processing(64byte)	0.8
ECU Update Module UDS Service processing	0.2
ECU Update Module EEPROM Write processing	0.4
Timer interrupt delay	0.02

에서 접근한 두 가지 방법보다 효율적으로 할 수 있다. 그림 6은 제안하는 알고리즘을 적용한 EUM 모듈과 AUTOSAR 모듈과의 관계를 표현한 것이다. 그림 7는 제시한 알고리즘을 이용한 ECU 업데이트의 순서를 표현한 것이다.

Ⅳ. 성능평가

본 논문에서는 시뮬레이션을 통해 제안 알고리즘의 성능을 평가하였다. 한국전자통신연구원에서 만든 AUTOSAR 플랫폼[9]에서 측정한 값을 시뮬레이션 데이터로 사용하였다. AUTOSAR DCM을 이용한 부분은 아직 개발단계여서 데이터를 측정하지 못하였다. 본 논문에서는 DCM 모듈을 이용한 ECU 소프트웨어 업데이트 부분은 생략하고 소프트웨어 컴포넌트를 이용한 업데이트 방법과 제안 알고리즘을 적용한 업데이트 방법을 비교하였다. 표 1은 시뮬레이션에 필요한 각 모듈의 소요시간을 기

표 2. 시뮬레이션 결과
Table 2. Simulation Results

Size of update data	Proposed system	Standard AUTOSAR
8 byte	1.5 ms	10.86 ms
40 byte	7.54 ms	10.86 ms
64 byte	12.04 ms	20.86 ms
400 byte	75.5 ms	110.86 ms
800 byte	156 ms	220.86 ms

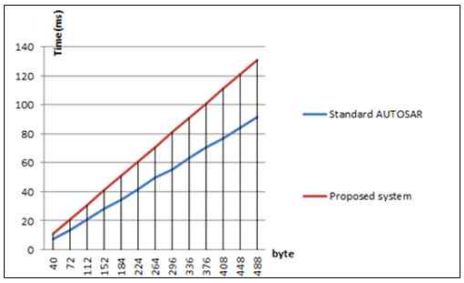

그림 8. SW 업데이트 소요시간
Fig. 8. Time of SW Update

록한 것이다.

AUTOSAR 표준의 ECU 업데이트를 하기위해서는 많은 모듈이 필요하다. 하지만 메모리의 저장속도 시험을 위해 반드시 필요한 모듈 이외에는 고려하지 않고 시뮬레이션 데이터를 산출하였다. 반드시 필요한 모듈에는 OS, GPT ISR, CAN ISR, 통신 관련 모듈과 메모리 관련 모듈이다. 그리고 RTE와 메모리 저장의 제어를 위한 Software Component 및 스케쥴 BSW TASK 만을 사용하였다. 실제 모든 용용소프트웨어가 동작을 하면 수행시간은 더 길어질 것이다. 제안된 알고리즘에서 사용한 모듈에는 GPT ISR, CAN ISR 그리고 확장된 ECU UPDATE 모듈만을 이용하여 데이터를 저장하기 위해 소요된 시간을 획득하였다. 표 2는 MATLAB을 사용하여 여러 용량의 ECU 업데이트 데이터를 EEPROM에 저장하는 두 가지 방식의 시뮬레이션 결과를 기록한 것이다.

표 1에서 얻은 파라미터 데이터로 표 2에서 보는 것과 같은 결과를 도출 할 수 있었다. AUTOSAR 표준 플랫폼에서 8 byte와 40 byte의 작업시간이 동일하게 걸리는 이유는 앞에서 언급하였지만 실제 EEPROM에 업데이트하는 시간은 Eep 메인 함수(Eep_MainFunction())이 동작하는 주기마

다 업데이트 하게 됨으로 10ms 주기로 약간의 데이터의 차이는 있지만 32~40byte 데이터를 저장하는 것을 시뮬레이션을 통해 알 수 있었다. 그림 8에서AUTOSAR 표준의 ECU 업데이트 저장하는 것을 최적화 하여 제안된 시스템과 비교한 것이다. 데이터의 크기에 따라 차이가 벌어지는 것을 확인 하였다.

Ⅴ. 결 론

AUTOSAR 플랫폼을 사용한 ECU가 증가함에 따라 ECU 소프트웨어 업데이트가 중요한 이슈가 되고 있다. 본 논문에서는 AUTOSAR 플랫폼 모듈 중에서 CAN If를 확장하여 ECU 업데이트 전용 모듈로 이동하고 즉시 메모리에 저장하는 방식을 제시하였다. 시뮬레이션 결과에서 기존 AUTOSAR 플랫폼 기반 소프트웨어 컴포넌트를 이용한 업데이트 방법은 적은 양의 데이터를 업데이트 할 때는 ECU 업데이트 시간이 길지 않았지만 대용량 업데이트에서는 매우 긴 시간이 소모되었다. 하지만 제안 알고리즘을 사용한 방법은 업데이트 데이터 사이즈가 증가할수록 기존 소프트웨어 컴포넌트를 이용하는 방법에 비해 업데이트 속도 측면에서 40% 이상의 성능향상을 보였다. 또한, 제안하는 알고리즘은 기존 AUTOSAR 소프트웨어 컴포넌트 사용방법과 DCM 사용방법에 비해 RAM 사용을 줄일 수 있다.

향후에는 제안된 알고리즘과 AUTOSAR DCM을 이용한 ECU 소프트웨어 업데이트 방법의 성능 비교를 수행하고 실제 ECU에서 제안 알고리즘을 동작시켜 실험을 통한 데이터 검증을 수행할 예정이다.

References

[1] K. Grimm, “Software Technology in an Automotive Company - Major Challenges,” Proceedings on International Conference of ICSE, pp.498-503, 2003.

[2] A. Pretschner, M. Broy, I.H. Kruger, T. Stauner, “Software Engineering for Automotive Systems: A Roadmap ,” Proceedings on FOSE, pp. 55-71, 2007.

[3] AUTOSAR homepage, http://autosar.org

[4] AUTOSAR GbR, Specification V4.1.0, 2010

[5] N. Navet, Y. Song, F. Simonot-Lion, C. Wilwert, “Trends in Automotive Communication Systems,” Proceedings on the IEEE, Vol. 93, No. 6, pp.1204-1223, 2005.

[6] AUTOSAR Consortium, Layered Software Architecture Ver.3.1.0, Release,4.0, Rev2, 2010

[7] AUTOSAR Consortium, List of Basic Software Modules Ver.1.5.0, Release,4.0, Rev2, 2010

[8] R. Machauer, “CAN configuration within Autosar,” Proceedings on the 11th International CAN Conference, 2006.

[9] E. Lee, J. Son, J. Kim, "Qplus-Auto : The Realtime Operating System based on AUTOSAR," Proceedings on Annual Conference of KSAE, 2010.

저 자 소 개

김 종 욱(Jong Uk Kim)

2006년 영남대학교 기계공학과 졸업.
2008년 영남대학교 전기공학과 석사.
현재, 한국전자통신연구원 선임연구원.

관심분야: 자동차전장시스템, Robotics
Email: jukim89@etri.re.kr

백 장 운 (Jang-Woon Baek)

2002년 경북대학교 자전기공학부 학사.
2004년 경북대학교 전자공학과 석사.
2009년 경북대학교 전자공학과 박사.

현재, 한국전자통신연구원 선임연구원.
관심분야: 차량전장용 SW, 센서 네트워크, 임베디드시스템.
Email: jwbaek98@etri.re.kr

권 기 구 (Kee-Koo Kwon)

2000년 경북대학교 전자공학과 석사.
2004년 경북대학교 전자공학과 박사.
현재, 한국전자통신연구원 선임연구원.

관심분야: 임베디드시스템, 자동차전장시스템, 영상신호처리.
Email: kwonkk@etri.re.kr

이 석 규 (Suk Gyu Lee)

1979년 서울대학교 전기공학과 학사.
1981년 서울대학교 전기공학과 대학원 석사. 1990년 UCLA 제어공학과 박사.

현재, 영남대학교 전기공학과 교수.
관심분야: Control Theory, Robotics.
Email: sglee@ynu.ac.kr