AUTOSAR기반 모터 응용 소프트웨어 컴포넌트 설계 방안 : EPS 시스템

주로 이벤트 입력방식으로 이루어지는 차량 바디 분야의 응용 시스템과는 달리 차량 모터 응용 시스템은 실시간 제어가 필요하기 때문에 모터 응용 소프트웨어를 설계하기 위해서는 다른 방법으로 접근해야 한다. 본 연구에서는 AUTOSAR (AUTomotive Open System Architecture) 표준 플랫폼을 이용하여 모터 응용 시스템 소프트웨어를 설계하는 방안을 소개하고자 한다. 대표적인 차량 모터 응용 시스템인 전동식 조향 시스템의 소프트웨어를 AUTOSAR 표준 규격에 준거하여 설계하고 시뮬레이션을 통해 이를 검증한다.

■ 박광민, 이성훈*
(대구경북과학기술원 미래산업융합기술연구부)

1. 서론

경제성, 환경성, 안정성, 편의성 및 운전 품질에 대해 끊임없이 요구하는 자동차 운전자들이 자동차 전장 시스템에서 전기제품의 성장을 견인하고 있다. 특히 자동차용 전기모터는 자동차 부품 중에서 가장 신장률이 높았고, 앞으로도 고도 신장할 것으로 예측되고 있다. 이는 자동차 대수의 증가보다는 자동차 한 대당 소요량의 증가로 해명된다. 현재, 자동차 1대당 사용하는 모터의 수는 50개에서 100개 정도이지만 일부 전장 메이커(OEM)에서는 자동차 전자화와 함께 전기모터의 비중이 커지면서 2015년에는 200개가 넘을 것으로 예측하고 있다[1].

예전에는 전기모터는 윈도우 리프트, 미러 제어 등의 차량 바디분야에 사용되는 DC 모터가 대부분이었지만, 차량에 전동식 조향 시스템, 능동 서스펜션, 제동 시스템, X-by-Wire 등의 새로운 응용 시스템들이 적용됨에 따라 AC 모터, BLDC 모터 등의 고성능 모터들의 수도 빠르게 증가하고 있다. 하지만 차량 모터 응용시스템의 비약적인 증가에 비해 이를 제어하는 응용 소프트웨어의 성능 및 안정성은 상대적으로 낮으며, 특히 AUTOSAR와 같은 차량 소프트웨어 표준 플랫폼으로 전이(migration)할 경우에는 소프트웨어의 설계 방안 조차 아직 미흡한 실정이다.

AUTOSAR는 하드웨어와 소프트웨어의 분리를 통하여 소프트웨어의 재사용성과 이식성을 보장하기 위하여 자동차 산업계에서 재정한 국제표준 소프트웨어 규격을 의미한다. AUTOSAR는 런타임 환경 하에 하드웨어와 소프트웨어 컴포넌트를 RTE (RunTime-Environment)라는 미들웨어를 통하여 분리시킴으로써, 소프트웨어의 재사용성과 신뢰성을 향상시킬 수 있다[2]. 하지만 AUTOSAR에서는 명세의 표준화에 따른 계산량과 데이터량이 늘어나기 때문에 마이크로 프로세서의 빠른 처리 속도가 요구되고 메모리의 요구사양이 높아져서 전체적으로 오버헤드가 커진다는 단점이 있다. 따라서 제한된 프로세서의 시간과 자원을 효율적으로 설계하고 분배하기 위해서는 개발대상인 응용 시스템의 특성과 목적에 맞게 소프트웨어 컴포넌트를 적절하게 설계하여야 한다. 특히 전동식 조향장치

(Electric Power Steering, EPS)와 같은 차량 모터 응용 시스템에서는 소프트웨어가 복잡하고 빠른 시스템 응답이 요구되기 때문에 응용 시스템의 특성을 정확히 파악하여 AUTOSAR 소프트웨어 컴포넌트(SoftWare Component, SWC) 아키텍처를 정의하고 런어블(runnable), 태스크(task) 등을 세부적으로 설계하는 것이 매우 중요하다.

본 연구에서는 대표적인 차량 모터 응용 시스템인 EPS 응용 소프트웨어에 대하여 빠른 시스템 응답 특성을 보장하면서 AUTOSAR 규격에 준거하여 설계하는 방법을 소개한다. 그리고 EPS 소프트웨어의 설계 과정을 효율적으로 검증하기 위한 AUTOSAR 기반의 SIL 테스트 환경을 소개하고자 한다.

2. AUTOSAR 개요

AUTOSAR는 응용 소프트웨어를 컴포넌트 단위로 개발하고 베이직 소프트웨어(Basic SoftWare, BSW)를 모듈화하여 설계된 표준 플랫폼으로 차량 임베디드 시스템의 재사용성을 높이고 기존의 하드웨어 종속적인 소프트웨어 구조에서 하드웨어 독립적인 구조로 바꾸는 것을 목표로 하고 있다 본 장에서는 AUTOSAR 소프트웨어 플랫폼 구조와 개발방법론에 대하여 간단히 소개한다.

2.1 AUTOSAR 소프트웨어 플랫폼

AUTOSAR 소프트웨어 구조는 그림 1과 같이 응용 소프트웨어 컴포넌트 계층, 미들웨어에 해당하는 RTE 계층, 베이직 소프트웨어 계층으로 구성되어 있다.

각 AUTOSAR 소프트웨어 컴포넌트는 응용 소프트웨어 기능의 일부를 구현하며, ECU에 맵핑되는 기본 단위이다. AUTOSAR 소프트웨어 컴포넌트는 AUTOSAR 인터페이스를 통하여 상호 데이터를 교환을 한다. AUTOSAR에서 제공하는 통신 인터페이스의 종류는 크게 두 가지로, 컴포넌트 간에 메시지를 주고받는 센더/리시버(sender/receiver) 통신 인터페이스와 함수 호출을 하는 클라이언트/서버(client/server) 통신 인터페이스 가 있다. AUTOSAR는 응용시스템을 구성하는 소프트웨어 컴포넌트들을 연결해 주는 VFB (Virtual Function Bus)라는 가상 버스 상에 컴포넌트들을 배치하고 이를 적절한 ECU에 할당하여 응용 시스템을 구성한다. 가상 버스를 제공하는 RTE 계층은 각 SWC 사이 및 SWC와 BSW 사이의 데이터 교환을 관장하는 통신센터의 역할을 하며, 동일한 인터페이스와 서비스를 제공한다. 따라서 응용 소프트웨어 컴포넌트 개발자가 하드웨어 종속적인 BSW 계층 및 네트워크 구성에 독립적으로 컴포넌트를 개발할 수 있는 환경을 제공함으로써 재사용성을 극대화하고 있다[3][4].

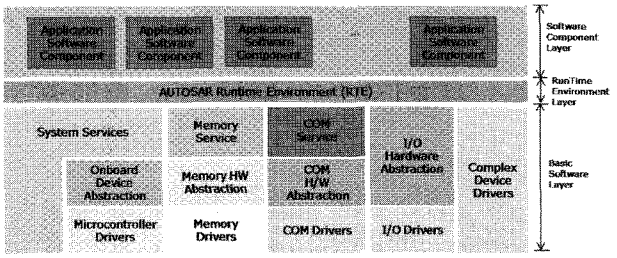


그림 1. AUTOSAR 소프트웨어 플랫폼의 계층 구조

2.2 AUTOSAR 개발 방법론

AUTOSAR는 기본적으로 컴포넌트 기반 소프트웨어 개발 방법론을 따르고 있으며 소프트웨어 개발 단계마다 해당 행위(activity)를 지원해주는 도구 및 플랫폼을 사용하고 있다. AUTOSAR 표준의 개발방법론은 크게 상위 설계에 해당하는 시스템 설계 단계와 ECU (Electric Control Unit) 설계 단계로 나누어진다. 시스템 단계에서는 소프트웨어 컴포넌트의 데이터 타입, 인터페이스 등을 정의하고 응용 소프트웨어 구현을 위한 런어블과 트리거 조건을 설계한다. 그리고 기본설계를 마친 소프트웨어 컴포넌트를 각 ECU에 맵핑하고 네트워크 설계를 함으로써 소프트웨어 컴포넌트 설계를 마친다. ECU 설계 단계에서는 System Configuration Description으로부터 각 ECU 정보를 추출하고 태스크 정의, RTE 및 BSW 등을 설계한다. 마지막으로 응용 소프트웨어, RTE, OS 등의 코드를 생성하고 컴파일, 링크를 수행하여 ECU Executable 환경을 구성한다[5].

3. AUTOSAR 기반 모터 응용 소프트웨어 설계

AUTOSAR 플랫폼에 맞게 차량 모터 응용 시스템의 소프트웨어를 설계하기 위해서는 이벤트 입력방식으로 주로 이루어지는 바디 분야의 응용 시스템과는 다른 방법으로 접근해야 한다. 모터의 제어부와 출력부는 보통 수 KHz나 수십 KHz의 높은 주파수로 구동되기 때문에 내부의 응용 소프트웨어 역시 높은 샘플링 주기로 정확한 타이밍에 제어가 이루어 져야 한다[6]. AUTOSAR 응용 소프트웨어를 구현하기 위해서는 소프트웨어 컴포넌트, 태스크, 런어블, RTE, 포트/인터페이스, Exclusive Area 등 새롭게 도입된 개념과 표준화된 구현 방식 등을 반영해서 설

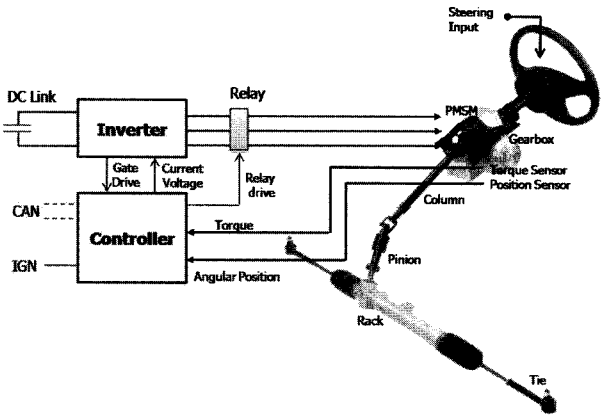


그림 2. 칼럼형 EPS 시스템의 구조

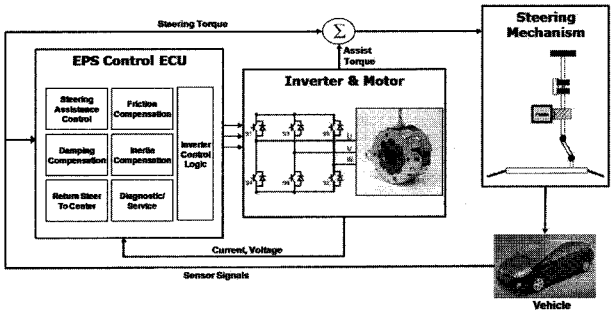


그림 3. EPS 시스템의 제어 블록다이어그램

계해야 한다. 따라서 모터 응용 소프트웨어를 설계하고 구현하기 위해서는 빠른 시스템 응답 요구사항을 고려하여 아키텍처 설계에서부터 스케줄링 설계까지 이루어져야 한다. 본 장에서는 차량 모터 응용 시스템의 소프트웨어 설계방안을 제시하기 위하여 전동식 조향 시스템의 소프트웨어를 AUTOSAR 규격에 준거하여 설계하는 방법을 소개한다.

3.1 EPS 응용 시스템

EPS 시스템은 차량의 속도, 조향 토크 등의 입력신호를 이용하여 운전자의 조향력을 보조해준다. 주차시와 같은 저속 주행 시에는 보조 토크의 크기를 크게 하여 쉽게 조향이 가능하도록 하고, 반대로 고속 주행 시에는 안전한 조향을 위하여 조향 핸들이 쉽게 돌아가지 않도록 보조 토크의 크기를 작게 제어한다. 또한 여러 가지 차량 센서신호를 이용하여 운전자의 조향 의도에 맞춰 능동 조향 제어까지 수행할 수 있다[7].

그림 2는 중소형 이하의 차량에 주로 사용되는 칼럼형 전동식 조향 시스템(C-EPS)의 구성도를 나타낸다. 본 EPS 시스템은 조향휠 입력부, 조향축 기구부, PMSM 모터부, 감속기어, 센서부, 그리고 모터 구동을 위한 ECU 제어부와 인버터 등으로 구성된다[8]. 운전자의 조향 입력 신호와 차량 속도 등의 입력 신호를 이용하여 전기 모터를 구동하고, 모터에 의해 생성된 토크는 컬럼 샤프트를 통해 피니언에 전달되어 운전자의 조향 토크를 보조하게 된다.

그림 3은 EPS 시스템의 제어 블록 다이어그램을 나타낸다. EPS 제어 시스템은 차량으로부터 속도 및 조향 토크값을 받고, 인버터로부터 전류, 전압 신호를 입력 받아서 운전자의 조향을 돕기 위한 최적의 제어를 수행하게 된다. EPS ECU는 차량 속도에 따라 보조 토크량을 제어하여 운전자의 조향 동력을 보조하는 것이 주기능이며, 이외에도 마찰력 보상 제어, 댐핑 보상 제어, 관성력 보상 제어, 조향 복원력 제어 등의 부가 기능까지 포함한다[8]. 제어된 신호는 인버터 제어 로직에서 좌표변환, 공간벡터제어 등을 수행함으로써 PMSM 모터 구동을 위한 고속의 PWM 신호로 변환된다. 마지막으로 모터의 출력 토크와 운전자의 조향 토크가 더해져서 조향 매커니즘부로 전달됨으로써 최종적인 EPS 시스템의 기능을 수행한다.

3.2 SWC 아키텍처 설계

차량용 응용 소프트웨어를 AUTOSAR 표준에 부합하도록 하기 위해서는 AUTOSAR 소프트웨어 컴포넌트 단위로 설계해야 한다. 따라서 기존의 개발된 응용 소프트웨어는 컴포넌트 단위로 재배치되고 컴포넌트내의 최소 실행단위인 런어블 단위로 모델링 되어야 한다. AUTOSAR Application Interface 규격문서에는 Steering Composition SWC 수준에서 인터페이스를 기술하고 있기 때문에, 내부의 Application SWC 인터페이스와 런어블과 태스크의 속성 등을 새롭게 설계해야 한다.

그림 4는 AUTOSAR 규격에 맞도록 설계된 EPS시스템의 소프트웨어 아키텍처이다[9]. EPS 소프트웨어는 크게 차량속도, 조향토크 등의 입력 신호를 처리하는 EPS Input SWC와 EPS의 동작여부를 담당하는 EPS Manager SWC, 입력신호로부터 전류 크기를 결정하고 모터를 제어하는 EPS Control SWC, 인버터를 구동하기 위한 최종 출력을 처리하는 EPS Motor Output SWC, 그리고 마지막으로 센서/액추에이터 컴포넌트가 마이크로 컨트롤러 추상화 계층(MicroController Abstraction Layer, MCAL)과 입출력을 주고 받기 위한 인터페이스를 제공하는 IO Hardware Abstraction SWC로 구성된다. 상위의 소프트웨어 컴포넌트는 다수 개의 Atomic 소프트웨어 컴포넌트로 구성되고 Atomic 소프트웨어 컴포넌트에는 한 개 이상의 런어블을 포함하고 있으며 총 20여 개의 런어블로 구성되어 있다.

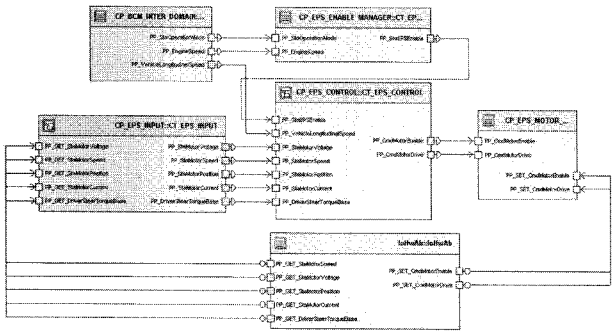


그림 4. EPS 소프트웨어 컴포넌트 아키텍처

3.3 EPS 소프트웨어 스케줄러 설계

AUTOSAR는 현재 사용되는 일반적인 스케줄링 방식과는 다른 방식으로 전체 아키텍처의 스케줄링이 이루어진다. 응용 소프트웨어는 구조적인 측면에서는 응용 소프트웨어 컴포넌트 단위로 구성되지만 스케줄링 측면에서 본다면 소프트웨어 컴포넌트 내부의 런어블과 AUTOSAR OS 스케줄러의 호출 단위인 태스크 단위로 재구성되고 맵핑된다[9].

그림 5는 EPS 시스템을 소프트웨어 컴포넌트 단위로 구성되는 구조적인 관점에서 보지 않고 런어블과 태스크의 스케줄링 관점에서 재구성한 SW 아키텍처를 나타낸다[10].

각각의 런어블들은 정적 타이밍 기반의 제어 알고리즘을 수행하기 위한 TimingEvent와 IO 하드웨어 추상화계층의 서버 컴포넌트와 인터페이스하기 위한 OperationInvokedEvent 트리거 방식으로 구동된다. AUTOSAR OS 스케줄러는 제어의 흐름에 의해 구분된 런어블들의 실행 순서와 주기를 결정하기 위해서 RTE에서 태스크 바디를 생성한다. 가장 단순하게 설계한다면 런어블 한 개당 하나의 태스크 바디를 생성하여 모든 태스크들을 정적 스케줄러 기반으로 설계할 수 있다. 하지만 이 방법은 OS의 태스크는 개수가 제한되어 있고 불필요한 태스크 스위칭에 의해 시간 낭비를 초래하기 때문에 적합하지 않다[11][12]. 런어블의 입출력 인터페이스와 실행 순서, 그리고 트리거 시간 주기 등을 고려하여 필수적인 태스크 바디만을 생성해야 한다. 하지만 반대로 태스크의 개수를 필요 이상 줄이게 되면 자칫 시스템의 성능에 영향을 줄 수 있기 때문에 최소의 태스크로 정확한 요구사항을 만족하도록 설계하는 것이 무엇보다 중요하다.

서버/클라이언트, 센더/리시버 입출력 인터페이스가 발생하는 부분에서 태스크 스위칭이 가장 많이 일어나기 때문에 해당 런어블들을 기본적으로 동일한 태스크에 맵핑한다. 하지만 제어 샘플링 주기의 차이가 심한 런어블은 다른 태스크로 맵핑한다. 그림 5에서 모터 구동과 출력에 관련된 런어블들은 샘플링

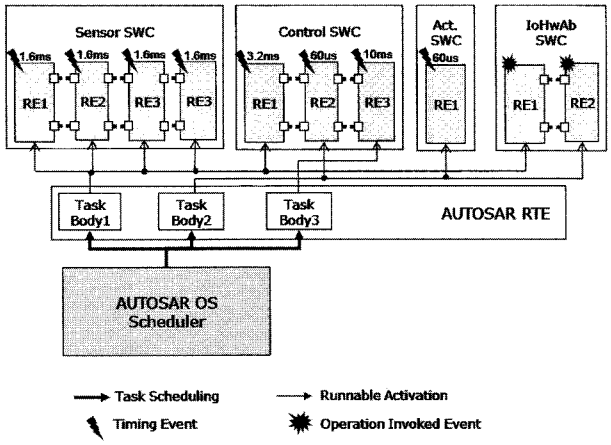


그림 5. EPS 소프트웨어 컴포넌트의 스케줄러 설계

Task 1 (Priority : 2, Preemption)

Runnable	Execution Order	Period	Activation Offset	GCD	Period for Sync.	Modified GCD
OP_GET for HMI Input 1	1	-	0	1.6ms	-	0.9ms
OP_GET for EPS Sensor Input 1	2	-	0		-	
OP_GET for EPS Sensor Input 2	3	-	0		-	
OP_GET for Vehicle Sensor Input 1	4	-	0		-	
HMI Input 1	5	1.6ms	0		0.9ms	
EPS Sensor Input 1	6	1.6ms	1.6ms		0.9ms	
EPS Sensor Input 2	7	1.6ms	3.2ms		0.9ms	
Vehicle Sensor Input 1	8	1.6ms	4.8ms		0.9ms	
EPS System Control	9	3.2ms	8.0ms		0.9ms or 1.8ms	

Task 2 (Priority : 3(High), Non-Preemption)

Runnable	Execution Order	Period	Activation Offset	GCD
EPS Motor Drive	1	60.0us	0	60.0us
Signal Output 1	2	60.0us	0	
OP_SET for Motor Enable Signal	3	-	0	
OP_SET for Motor Drive Signal	4	-	0	

Task 3 (Priority : 1(Low), Preemption)

Runnable	Execution Order	Period	Activation Offset	GCD	Period for Sync.	Modified GCD
EPS Enable Manager	1	10ms	0	10ms	9ms	9ms

그림 6. EPS 소프트웨어 스케줄링 상세 설계 명세

주기가 60.0us으로 다른 런어블들에 비해 수십배 이상 차이가 나기 때문에 태스크 바디를 따로 생성하여 할당해야 한다. 그리고 10ms의 샘플링 주기를 가지고 있는 런어블 역시 다른 런어블들에 비해 샘플링 주기가 상이하기 때문에 다른 태스크로 할당한다. 하지만 태스크 단위의 스케줄링이 필요 없는 경우라면 태스크 스위칭 시간을 최소화하기 위해 태스크 1과 태스크 3을 하나의 태스크로 통합하는 것이 좋다.

스케줄러의 개념 설계를 마친 후, 태스크 개수와 지연 시간를 최소화하고 태스크 간의 시간 동기화를 고려하여 런어블의 실행 순서, 오프셋 시간, 태스크의 최대주기(Greatest Common Divisor, GCD) 등의 세부 속성을 정의한다. 그림 6은 EPS 소프트웨어 컴포넌트의 스케줄러의 상세 설계 테이블을 나타낸다.

3.4 ECU설계

시스템 설계가 완료된 후, ECU 별로 소프트웨어 통합 작업을 수행한다. 각 응용 소프트웨어 컴포넌트와 RTE, 그리고 BSW 부분을 환경 설정 및 코드 생성 과정을 거친 후에 통합하게 된다. BSW 부분은 내부 구성단위에 따라 ICC1 (Implementation Conformance Class 1), ICC2, ICC3으로 구분된다. ICC1은 BSW 전체를 하나의 클러스터로 설계하고, ICC2는 BSW를 기능별 클러스터 단위로 구성하여 설계한다. 그리고 ICC3은 세분화된 모듈 단위로 구성된다. 본 연구에서는 가상의 ECU인 PC-Target으로 BSW를 환경 설정하였으며, ICC1으로 설계하여 적용하였다[7].

4. 시험 및 결과

4.1 시험 환경

AUTOSAR 플랫폼을 적용하여 ECU 설계가 완료된 후, PC 환경에서 가상의 ECU를 생성하여 설계된 EPS의 기능을 시뮬레이션 및 테스트 할 수 있다. SIL (Software In the Loop) 환경을 구성할 때, 실제 환경과 유사한 시험 결과를 얻기 위해서는 EPS의 모터, 기구부 등이 포함된 플랜트와 액추에이터를 상세하고 정확하게 구현해야 한다. 바디 분야는 통상적으로 상태 다이어그램으로 구현된 로직으로 입출력을 ON/OFF 제어하므로, 플랜트 부분이 실험 결과에 미치는 영향이 적다. 하지만 실시간 고속 제어가 필요한 섀시 분야에서는 정확하고 상세한 플랜트 모델을 이용해야만 설계된 알고리즘의 성능을 제대로 평가할 수 있다. 따라서 EPS 제어 로직을 제외한 기구부는 MATLAB의 SimPowerSystem 블록셋에 있는 모터, 인버터 모델을 기반으로 EPS 매커니즘, 그리고 운전자 모델까지 설계한다. MATLAB은 높은 스위칭 주파수로 Fixed Point 연산이 가능하고, AUTOSAR 플랫폼과의 인터페이스를 지원하기 때문에 본 시스템을 테스트하기 위하여 보다 적합한 환경을 제공한다. 그림 7은 EPS의 SIL 시험을 위한 전체 시뮬레이션 환경을 보여준다. AUTOSAR 기반으로 설계된 EPS 소프트웨어 컴포넌트로부터 생성된 DLL 파일으로 부터 가상의 EPS ECU를 생성하여 AUTOSAR 기반 차량 네트워크 시뮬레이션 도구인 CANoe에서 노드로 구성한다. EPS 시스템에서 모터와 인버터를 포함한 출력 컴포넌트는 빠른 실행속도와 고속의 샘플링 주기를 지원하는 MATLAB 환경으로 구성하여 실험하였다. 그리고 시뮬레이션 도구 간의 입출력 데이터 공유 및 실시간 시간 동기화를 위하여 CANoe에서는 MATLAB 연동을 위한 환경 변수를 추가로 생성하고

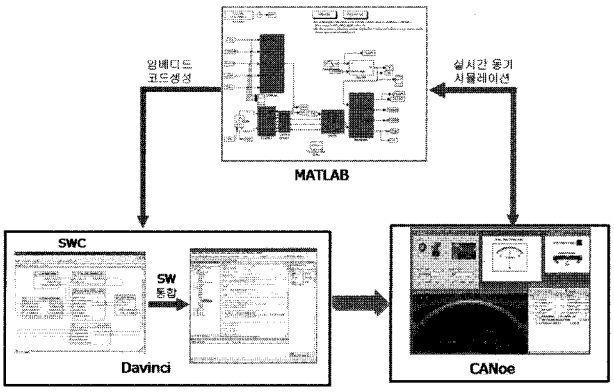
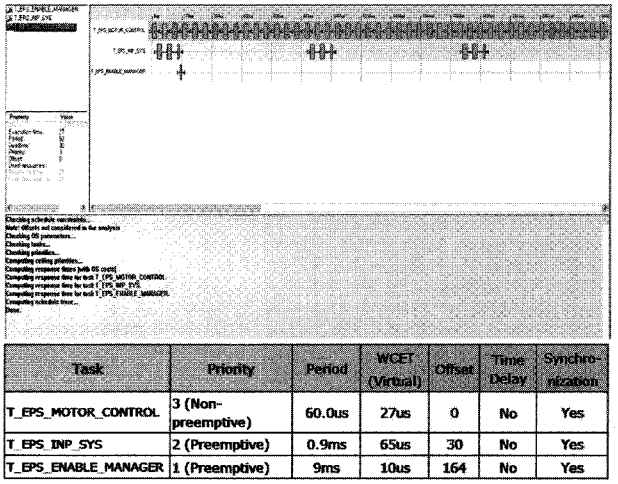


그림 7. EPS SIL 시스템 개발환경



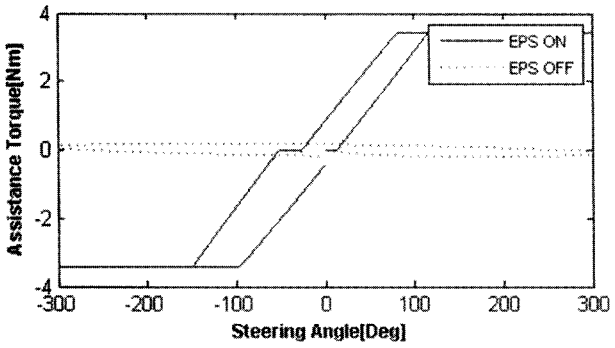
Task	Priority	Period	WCET (Virtual)	Offset	Time Delay	Synchro-nization
T_EPS_MOTOR_CONTROL	3 (Non-preemptive)	60.0us	27us	0	No	Yes
T_EPS_INP_SYS	2 (Preemptive)	0.9ms	65us	30	No	Yes
T_EPS_ENABLE_MANAGER	1 (Preemptive)	9ms	10us	164	No	Yes

그림 8. EPS 시스템의 타이밍 검증

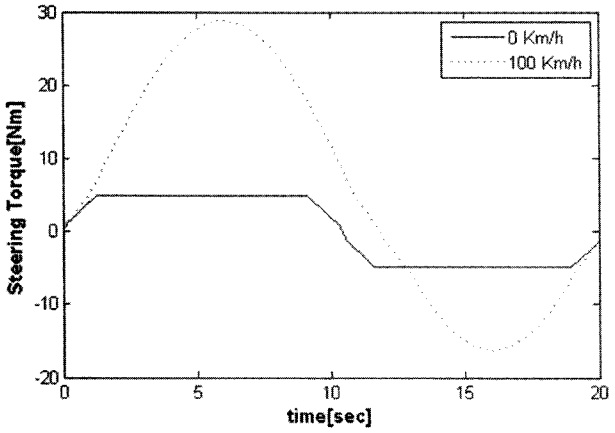
MATLAB에서는 Simulink에 있는 CANoe S-Function 블록셋을 사용하여 실시간 동기 시뮬레이션(Concurrent-Simulation)환경을 구성하였다.

4.2 타이밍 시뮬레이션 결과

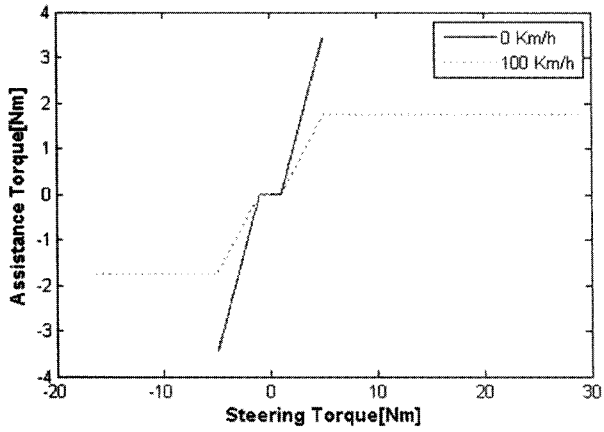
설계된 EPS 소프트웨어 컴포넌트를 태스크 기반의 OS 타이밍 분석 도구에서 타이밍 시뮬레이션을 수행하였다. 그림 8과 같이 샘플링 주파수를 일정하게 유지하면서 태스크 간의 시간 동기화가 이루어지는 것을 확인하였다. 또한 지연시간이 없고 데드라인 이내로 수행되는 등 대부분의 시간 요구사항을 만족하였다. 모터 구동 태스크의 데드라인(deadline)은 모터구동부의 최악응답시간이 샘플링 주기의 50% 이하가 되어야 한다는 가정 아래 30us으로 설정하였다. 그 외 태스크의 실행시간은 런어블 수와 코드라인 등으로 추정한 가상의 값이고, 지터 시간과 외부 인터럽트 등은 고려하지 않았다.



(a)



(b)



(c)

그림 9. EPS 기능시험 결과

4.3 기능 시험 결과

설계된 EPS ECU의 성능을 평가하기 위해 구축한 SIL 개발환경에서 테스트를 수행하였다. 그림 9 (a)는 운전자가 조향각 입력을 가하였을 경우, 모터의 보조 토크의 크기를 EPS의 동작 유무에 따라 나타낸 그래프이다. EPS ON의 히스테리시스 곡선에서 정방향과 역방향 사이의 토크 오프셋이 적을수록 조향감과 복원력 제어가 우수하다고 할 수 있다. 그림8 (b)와 (c)는 제어기 기능 시험을 위하여 차량 정차 상태(0 Km/h)와 고속 주행상태(100 Km/h)에 대하여 운전자 조향토크와 보조토크의 값을 각각 비교하였다. 모터의 보조토크는 인가된 차량의 속도와 운전자의 조향 토크에 따라 미리 설계된 조향 토크맵을 기준으로 출력되었고, 비교적 토크 리플이 작은 것을 확인할 수 있다.

5. 결론

현재 차량의 전자화 추세에 따라 자동차 전장에 사용되는 전기모터의 수도 급격하게 늘어나고 있다. 하지만 이에 비해 모터 응용 시스템을 제어하는 응용 소프트웨어는 대부분 복잡한 레거시(legacy) 코드로 구현되어 있어서, AUTOSAR 차량 소프트웨어 표준 플랫폼에 맞게 재설계하기 위해서는 많은 어려움이 있다.

AUTOSAR 응용 소프트웨어를 구현하기 위해서는 소프트웨어 컴포넌트, 태스크, 런어블, RTE 등 새롭게 도입된 개념과 표준화된 구현 방식 등을 충분히 반영해서 설계해야 한다. 특히 빠른 시스템 응답성이 요구되는 차량 모터 응용 시스템에서는 제한된 프로세스의 시간과 자원을 효율적으로 사용하기 위해서 AUTOSAR 아키텍처 설계에서부터 스케줄링 설계까지 적절하게 이루어져야 한다.

본 연구에서는 AUTOSAR 표준 플랫폼을 이용하여 모터 응용 시스템 소프트웨어를 설계하는 방안을 소개하였다. 대표적인 차량 모터 응용 시스템인 EPS 시스템에 대하여 소프트웨어 컴포넌트를 설계하고 새시 시스템에 적합하게 런어블, 태스크의 스케줄링 설계를 하였다. 그리고 PC기반의 실시간 동기 시뮬레이션 환경을 구축하여 AUTOSAR 표준에 맞게 설계된 EPS 소프트웨어 컴포넌트를 검증하였다. 하지만 보다 신뢰성 있는 차량 ECU 소프트웨어를 개발하고 검증하기 위해서는 정확한 요구사항과 실차 기반의 테스트가 필요하다.

참고문헌

[1] A&D컨설턴트, "자동차용 모터 시장과 기술동향," 2008년도 특별보고서, pp. 13~37, 2008.

[2] http://www.autosar.org

[3] AUTOSAR "Layered Software Architecture," AUTOSAR Specification R.3.1, 2009.

[4] AUTOSAR "Interaction with Behavioral Models," AUTOSAR Specification R.3.1, 2009.

[5] W. Oh and J. Shin, "Model Based Development Process for the Embedded System in Vehicle," *SAE World Congress*, 2008.

[6] Ji-Hoon Kim and Jae-Bok Song, "Control logic for an electric power steering system using assist motor," *Mechatronics 12*, p.447-459, Nov. 2002.

[7] G. Park, S. Lee, and D. Kum, "Development of Software Component for EPS System based on AUTOSAR," *KSAE 2009 Annual Conference*, 2009.

[8] J. Kim and J. Song, "Control logic for an electric power steering system using assist motor," *Mechatronics 12*, pp. 447-459, Nov. 2002.

[9] R. Racu, K. Richter, and R. Ernst, "The Need of a Timing Model for the AUTOSAR Software Standard," *Workshop on Models and Analysis Methods for Automotive Systems (RTSS Conference)*, 2006.

[9] AUTOSAR "Explanation of application interfaces of the chassis domain," AUTOSAR Specification R.3.1, 2009.

[10] A. Ferrari and M. D. Natale, "Time and memory tradeoffs in the implementation of AUTOSAR components," *EDAA 2009 Conference*, 2009.

[11] A. Burns, K. Tindell, and A. Wellings, "Effective Analysis for Engineering Real-Time Fixed Priority Schedulers," *IEEE Transactions on Software Engineering*, vol. 21, no. 5, pp. 475-480, 1995.

[12] K. Richter and M. Jersak, "Scheduling Analysis and Optimization for Safety-Critical Automotive Systems," *SAE 2008 World Congress*, 2008.

저자약력

박 광 민

· 2005년 한국항공대 항공기계학과 학사.
· 2007년 광주과학기술원 기계전자과 석사.
· 2007년~현재 대구경북과학기술원 연구원.
· 관심분야: 차량 임베디드 시스템, AUTOSAR.

이 성 훈

· 1996년 경북대학교 전자공학과 학사.
· 1998년 경북대학교 전자공학과 석사.
· 2007년 경북대학교 전자공학과 박사.
· 1999년~2002년 대우정밀 기술연구소.
· 2002년~2005년 국방과학연구소.
· 2005년~현재 대구경북과학기술원 선임연구원.
· 관심분야: 차량 임베디드 시스템.