# AUTOSAR 기반 EPS 시스템 소프트웨어 컴포넌트의 스케줄링 설계 및 시뮬레이션

# Scheduling Design and Simulation of Software Components for EPS System based on AUTOSAR

박 광 민, 금 대 현, 손 병 점, 이 성 훈*
(Gwangmin Park, Daehyun Kum, Byeongjeom Son, and Seonghun Lee)

**Abstract:** Through the AUTOSAR methodology, the embedded software shall become more flexible, reusable, maintainable than ever. However, it is not mentioned about specific timing constraints of software components in AUTOSAR. There are a few basic principles for mapping runnable entities. At this point, AUTOSAR software design with optimal scheduling method is one of the enabling technologies in vehicle embedded system. This paper presents an approach based on mapping runnable entities and task scheduling design method for EPS (Electric Power Steering) software components, based on AUTOSAR. In addition, the experimental results of concurrent simulation show that the proposed scheduling technique and timing synchronization in the software component design can achieve the improved torque ripple performance and it well suited for EPS application software.

**Keywords:** EPS (Electric Power Steering), AUTOSAR, SWC (Software Component), scheduling, timing analysis, concurrent-simulation

## I. 서론

최근 차량의 전자화가 빠르게 진전되면서 차량 소프트웨어의 개발 규모가 비약적으로 커졌다. 또한 무인 자동차, 전기 자동차 등 미래형 자동차의 연구 개발과 맞물려 차량 임베디드 소프트웨어는 앞으로 더욱 복잡해 질 것으로 예측된다. 이러한 복잡한 소프트웨어를 빠르고 신뢰성 있게 개발하기 위하여 AUTOSAR (AUTomotive Open System Architecture)는 차량용 소프트웨어 아키텍처와 개발방법론 등의 표준 명세를 정의하였다. AUTOSAR는 런타임 환경 하에 하드웨어와 소프트웨어 컴포넌트를 RTE (RunTime-Environment)라는 미들웨어를 통하여 분리시킴으로써, 소프트웨어의 재사용성과 신뢰성을 향상시킬 수 있다[1].

하지만 AUTOSAR에서는 명세의 표준화에 따른 계산량과 데이터량이 늘어나기 때문에 전체적으로 오버헤드가 커진다는 단점이 있다. 이로 인해 마이크로 프로세스의 처리 속도를 높이고 ROM/RAM 용량 또한 늘려야 하므로, 결국 이것은 가격 상승으로 이어지게 된다. 더욱이 AUTOSAR에서는 운영체제가 필수 컴포넌트이기 때문에 이 또한 오버헤드가 증가하는 원인이 될 수 있다. 따라서 제한된 프로세스의 시간과 자원을 효율적으로 설계하고 분배하기 위해서는 개발 대상인 응용 시스템의 특성과 목적에 맞게 소프트웨어 컴포넌트를 적절하게 스케줄링 하여야 한다. 특히 차량 바디나 파워트레인 도메인보다 빠른 시스템 응답이 요구되는 차량 섀시 시스템과 같은 실시간 분산제어 시스템에서는 시간에 대한 요구사항을 만족하지 못한다면 기능 상의 오류뿐만 아니라 사고로 연결될 수 있으므로, 타이밍을 예측하고 분석하는 것이 보다 중요한 문제이다.

AUTOSAR 소프트웨어 아키텍처는 복잡한 구조를 가지고 있기 때문에 기존 방식을 그대로 적용하여 타이밍을 분석하고 예측하는 것은 어렵다. AUTOSAR 응용 소프트웨어는 소프트웨어 컴포넌트, 런어블(runnable), 태스크(task) 간의 다양한 인터페이스 방식과 상관관계에 의해서 타이밍이 결정된다. 그러나 이러한 복잡한 타이밍 설계구조를 가지고 있음에도 불구하고 AUTOSAR에서는 응용 소프트웨어 컴포넌트의 타이밍 모델과 스케줄링 관련된 내용을 거의 제시하지 않기 때문에, 개발자가 설계 단계에서부터 시간 검증과 스케줄링 설계를 수행하는 것이 용이하지 않다[2]. 이것은 결국 소프트웨어 컴포넌트의 통합을 어렵게 만들고, 스케줄링 분석과 타이밍 검증은 모든 컴포넌트들이 태스크와 ECU에 할당된 이후에나 비로소 가능하게 된다. 따라서 AUTOSAR 소프트웨어 아키텍처의 정확한 타이밍 설계 및 분석을 위해서는 기존의 스케줄링 방식과는 차별화된 AUTOSAR 스케줄링 전략이 필요하다.

EPS 시스템은 실시간 차량 정보에 따라 적절하게 전기모터를 제어하여 운전자의 조타력을 보조해 주는 장치로 빠른 스위칭 주파수의 제어가 요구되는 섀시 제어 시스템이다. EPS 시스템에서 충분한 성능과 빠른 시간 응답 특성을 보장하기 위해서는 정확한 플랜트 모델링, 제어기의 최적 설계뿐만 아니라 높은 스위칭 주파수 출력을 보장하기 위한 정확한 타이밍 해석과 스케줄링이 필요하다.

본 논문에서는 AUTOSAR 기반의 EPS 소프트웨어 컴포넌트의 타이밍 분석 및 동기화, 그리고 최적 스케줄링을 하기 위한 런어블, 태스크의 스케줄링 설계방법을 제안한다. 그리고 PC 기반의 실시간 동기 시뮬레이션을 통해 제안한 방법의 타당성을 확인한다.

* 책임저자(Corresponding Author)
논문접수: 2010. 3. 15., 수정: 2010. 4. 15., 채택확정: 2010. 4. 30.
박광민, 금대현, 손병점, 이성훈: 대구경북과학기술원 미래산업융합기술연구부
(ggangmin@dgist.ac.kr/kumdh@dgist.ac.kr/easypass@dgist.ac.kr/shunlee@dgist.ac.kr)
※ 본 연구는 교육과학기술부에서 지원하는 대구경북과학기술원의 기관고유사업비로 수행하였음.

## II. AUTOSAR 스케줄링

### 1. AUTOSAR 아키텍처의 스케줄링 구조

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.

그림 1에서와 같이 AUTOSAR OS 스케줄러는 기본적으로 소프트웨어 컴포넌트와 베이직 소프트웨어를 태스크 단위로 스케줄링 한다. 소프트웨어 컴포넌트는 기능을 수행하는 최소 단위인 런어블들로 구성되며, 베이직 소프트웨어는 내부의 BSW Scheduled 함수 단위로 구성된다[3,4]. 그리고 다시 런어블은 RTE (Run-Time Environment)에 의해 태스크 단위로 맵핑되고 스케줄링 속성이 정의되며, BSW Scheduled 함수는 BSW 스케줄러에 의해 태스크 단위로 스케줄링 속성이 정의된다. AUTOSAR R3.1기준으로 상단의 소프트웨어 컴포넌트 계층과 하단의 베이직 소프트웨어 계층은 RTE에 의해서 타이밍 속성이 완전히 분리되어 동작한다.

### 2. AUTOSAR 소프트웨어 컴포넌트 스케줄링

#### 2.1 AUTOSAR 태스크, 런어블

AUTOSAR 소프트웨어 컴포넌트는 기능을 수행하는 최소 단위인 런어블들로 구성되며, 응용 소프트웨어 컴포넌트의 스케줄링은 런어블을 통해서 이루어진다. 런어블 실행순서, RTE 이벤트의 트리거 조건, 트리거 주기 등의 기본 속성들이 정의되면 OS 스케줄러의 실행 단위인 태스크에 맵핑하게 된다. OS 스케줄러는 미리 정의된 태스크 속성과 스케줄 테이블에 따라 태스크를 구동하고 태스크 내부에 맵핑된 하위 런어블들을 실행한다.

#### 2.2 RTE 스케줄링

RTE는 소프트웨어 컴포넌트의 스케줄링과 관련하여 다음과 같은 세 가지의 중요한 기능을 수행한다.

첫째, RTE는 런어블을 트리거 하기 위한 RTE 이벤트 방식을 결정한다. RTE는 TimingEvent, DataReceivedEvent, DataReceivedErrorEvent, OperationInvokedEvent 등 7여 가지의 이벤트 타입을 지원한다. 이 중 정적 타이밍 스케줄링에 해당하는 TimingEvent는 OS alarm에 의해 구현되고 나머지는 대부분 AUTOSAR OS 이벤트에 의해 직접 구현된다.

둘째, RTE는 태스크 바디를 생성하여 OS 스케줄러가 런어블을 호출하기 위한 중재자의 역할을 수행한다. 기능 단위의 런어블이 BSW 계층에 해당하는 OS 스케줄러에 의해 구동되기 위해서는 RTE에서 태스크 바디를 생성하여 런어블을 호출해야 한다.

셋째, RTE는 SetEvent()나 ActivationTask(), TerminateTask() 와 같은 OS 서비스를 호출함으로써 태스크의 활성화(Activation) · 재개(Resumption) · 종료(Termination) 등을 제어한다[5,6].

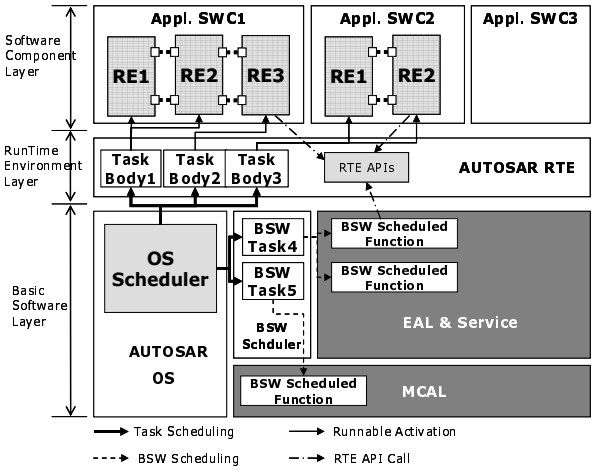



그림 1. AUTOSAR 소프트웨어 아키텍처의 스케줄링 구조.

Fig. 1. Scheduling structure of AUTOSAR software architecture.

## III. EPS 시스템

### 1. EPS 시스템 소개

EPS 시스템은 조향각, 조향 토크, 차량의 속도 및 가속도 등의 차량 센서 정보에 따라 조향 동작을 위한 보조토크를 효과적으로 제어하여 운전상황에 따라 조향 부하를 줄이고 조타력을 보조해 주는 장치이다[7]. EPS 시스템은 그림 2와 같이 차량 속도에 따라 보조 토크량을 제어하여 운전자의 조향 동력을 보조하는 것이 주기능이며, 이외에도 댐핑 보상 제어나 마찰력 보상 제어 등의 부가 기능까지 포함한다[8].

### 2. EPS 시스템 구조

그림 3은 중소형 이하의 차량에 주로 사용되는 칼럼형 전동식 조향 장치(C-EPS)의 구성도를 나타낸다. 본 EPS 시스템은 조향휠 입력부, 조향축 기구부, PMSM 모터부, 감속기어,

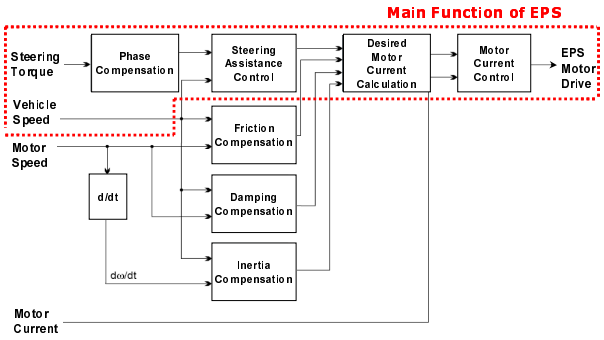



그림 2. EPS 시스템의 기능 블록도.

Fig. 2. Functional block diagram of EPS system.

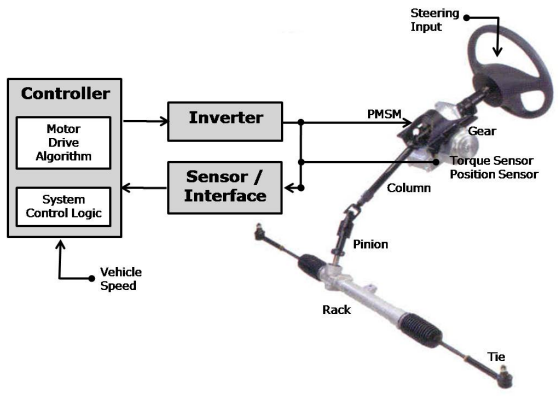



그림 3. 칼럼형 EPS 시스템의 구성도.

Fig. 3. Configuration diagram of column assist Type EPS system.

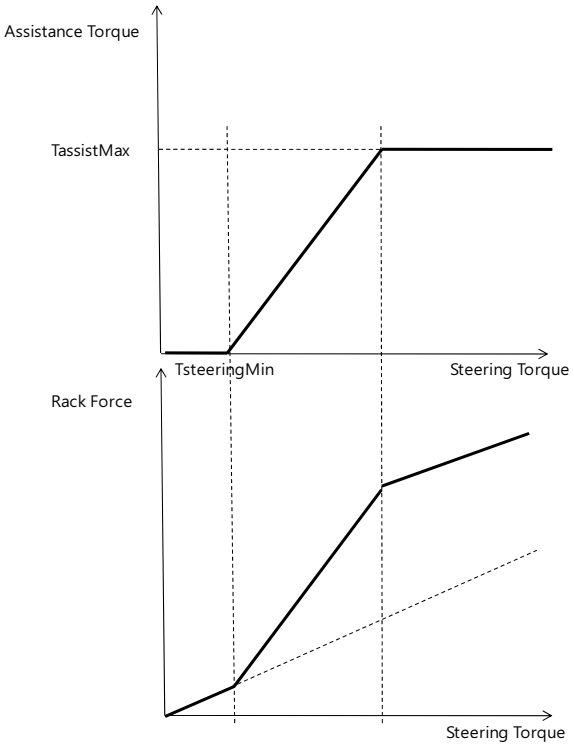



그림 4. 조향 토크 vs. 보조 토크 vs. 랙반력 그래프.
Fig. 4. Steering Torque vs. Assistance Torque vs. Rack Force Graph.

센서부, 그리고 모터 구동을 위한 ECU 제어부와 인버터 등으로 구성된다. 운전자의 조향력은 조향축 기구부에 장착된 토크센서로부터 측정되고 차량 속도 신호는 ECU 외부의 통신에 의해 인가된다. 운전자 토크와 차속, 그리고 다른 시스템 변수로부터 조향 보조력의 크기와 방향을 결정되고, 모터 구동 알고리즘과 인버터, 모터로 이루어지는 구동부에 의하여 토크제어가 적절하게 수행된다[9].

### 3. EPS 시스템 요구사항

EPS의 주기능인 조향 보조 토크제어만을 고려하면, EPS 시스템의 기능 요구사항은 기본적으로 저속 주행 시에는 보조토크를 크게 하고 고속 주행 시에는 보조 토크를 작게 하여 운전자의 조향력을 보조하는 것이다. 그림 4와 같이 운전자의 조향 토크가 EPS 동작을 하기 위한 최소값 TsteeringMin 이상부터 보조 토크가 인가되기 시작하여 보조 토크의 최대값에 근접할 때까지 선형적으로 증가한다. TsteeringMin과 TassistMax사이에서는 랙반력이 증가할 수 있도록 선정하고 랙반력과 조향 토크의 차이만큼을 모터의 보조 토크로 인가한다. 따라서 EPS 시스템은 차량 속도가 증가함에 따라 보조 토크의 크기를 줄이고 임계값인 TassistMax도 줄여서 운전자에게 속도 감응형 특성을 제공한다[8]. 또한 조향각과 조향 보조토크 간의 히스테리시스 곡선을 결정함으로써 조향 복귀력과 조향감도에 대한 요구사항을 정의하고 평가할 수 있다.

조향 토크 특성곡선에서 토크 리플을 적게 하고 빠른 시간 응답 특성이 보장하기 위해서는 플랜트 모델링, 제어기 설계뿐만 아니라 출력 주파수를 보장하기 위한 정확한 타이밍 해석과 스케줄링이 필요하다.

### 4. EPS 시스템 소프트웨어 컴포넌트

EPS 시스템의 기능을 AUTOSAR 소프트웨어로 구현하려면 각 기능별로 소프트웨어 컴포넌트를 나누고 AUTOSAR 표준에 기반하여 포트 인터페이스, 데이터 타입, 런어블 등을 설계해야 한다. 그림 5는 AUTOSAR Application Interface 규격 문서를 기반으로 설계한 EPS 시스템의 소프트웨어 아키텍처이다[10]. EPS 소프트웨어는 크게 차량속도, 조향토크 등의 입력 신호를 처리하는 EPS Input SWC와 EPS의 동작여부를 담당하는 EPS Manager SWC, 입력신호로부터 전류크기를 결정하고 모터를 제어하는 EPS Control SWC, 인버터를 구동하기 위한 최종 출력을 처리하는 EPS Motor Output SWC, 그리

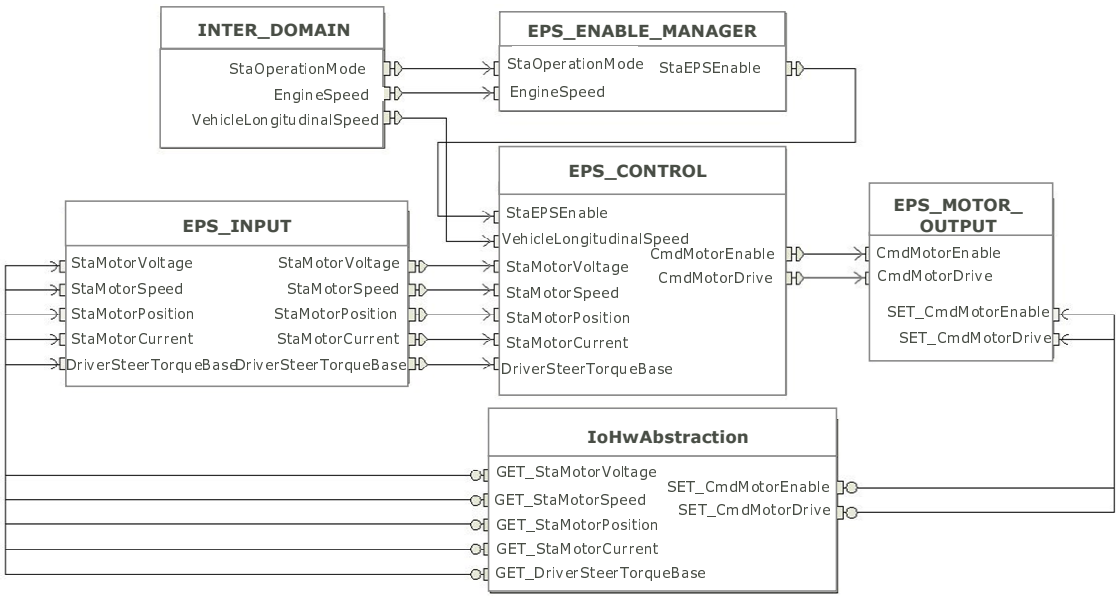



그림 5. EPS 시스템 소프트웨어 아키텍처.
Fig. 5. Software Architecture of EPS System.

고 마지막으로 센서/액추에이터 컴포넌트가 마이크로 컨트롤러 추상화 계층(MCAL)과 입출력을 주고 받기 위한 인터페이스를 제공하는 IO Hardware Abstraction SWC로 구성된다. 각 소프트웨어 컴포넌트에는 한 개 이상의 런어블을 포함하고 있으며 총 20여 개의 런어블로 구성되어 있다.

## IV. EPS 소프트웨어 컴포넌트의 스케줄링 방법

EPS 섀시 제어 시스템에서 토크 리플을 적게 하고 빠른 시간 응답 특성이 보장하기 위해서는 정확한 플랜트 모델링, 제어기의 최적설계 뿐만 아니라 높은 스위칭 주파수 출력을 보장하기 위한 정확한 타이밍 해석과 스케줄링이 필요하다. 본 장에서는 AUTOSAR 표준 플랫폼에 기반한 EPS 소프트웨어 컴포넌트의 스케줄링 방법을 제시한다.

### 1. EPS 시스템 스케줄링 방안

모터와 인버터 등으로 이루어 지는 EPS 제어 시스템의 경우 모터의 출력 주파수가 빠르고, 입력 및 제어 신호가 정확한 타이밍에 인가되어서 원하는 시간 요구사항을 만족해야 하므로 섀시 제어 시스템의 특성에 맞는 스케줄링 전략이 필요하다. EPS 시스템은 정적 타이밍 기반의 실시간 분산제어 시스템이므로 입력 주기가 불규칙하고 시간 검증이 어려운 이벤트 트리거 방식보다는 시간 트리거 방식이 보다 적합하다[11]. 또한 출력과 관련된 태스크는 안전과 직결되기 때문에 가장 높은 우선순위로 설계하거나 다른 태스크에 의해 선점되지 않는 비선점(non-preemptive) 스케줄링 방식을 사용하는 것이 효율적이다. 출력 태스크를 제외한 나머지 태스크들은 선점형(preemptive) 고정 우선순위 스케줄링 방식을 사용하여 AUTOSAR OS 스케줄러를 설계한다.

또한 빠른 출력 주파수를 확보하기 위해서는 시간 지연을 최소화하고 태스크의 스위칭 횟수를 줄여서 태스크의 최악 응답시간을 줄이는 것도 중요하다. 그리고 데이터 일관성(data consistency)을 유지하고 오버샘플링(over-sampling)과 언더샘플링(under-sampling)을 최소화하기 위해서는 시간 동기화를 고려하여 런어블과 태스크의 타이밍을 설계해야 한다.

### 2. 태스크/런어블 설계 방법

섀시 제어 시스템은 기본적으로 센서입력 컴포넌트, 시스템 제어 컴포넌트, 액추에이터 출력 컴포넌트와 같은 구조로 응용 소프트웨어가 설계된다. EPS 시스템도 마찬가지로 제어의 방향으로 볼 때 엔코더나 홀센서 등의 센서 입력을 받아서 처리하는 센서 컴포넌트, 센서 입력과 레퍼런스 입력으로부터 미리 정의된 알고리즘을 처리하는 제어 컴포넌트, 제어된 출력신호로부터 모터를 구동하는 액추에이터 컴포넌트로 구성되기 때문에 일련의 신호 전달 방향을 예측할 수 있다. 제어 방향과 동일하게 런어블의 실행순서를 정의하고 런어블의 타이밍을 결정하고 트리거 타이밍을 동기화 시켜 줌으로써 시간 낭비를 최소화하고 오버샘플링(over-sampling)과 언더샘플링(under-sampling)을 최소화할 수 있다.

그림 6은 EPS 시스템을 소프트웨어 컴포넌트 단위로 구성되는 구조적인 관점에서 보지 않고 스케줄링 관점에서 런어블과 태스크 단위로 재구성한 SW 아키텍처를 나타낸다[12].

각각의 런어블들은 정적 타이밍 기반의 제어 알고리즘을 수행하기 위한 TimingEvent와 IO HW 추상화계층의 서버 컴포넌트와 인터페이스하기 위한 OperationInvokedEvent 트리거

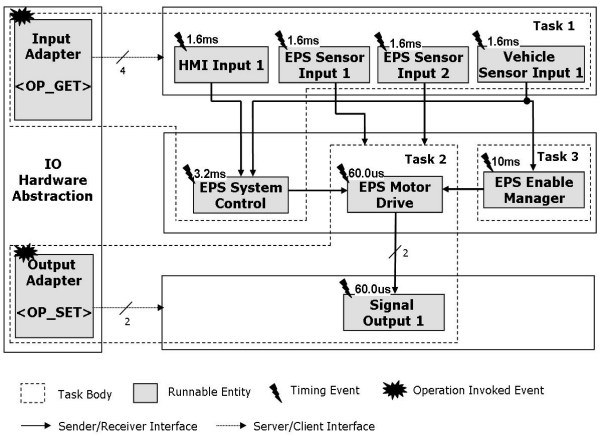



그림 6. 스케줄링 기반 EPS 소프트웨어 구조.

Fig. 6. EPS software structure based on scheduling concept.

방식으로 구동된다. 초기화를 담당하는 런어블들을 제외해도 IO 하드웨어 추상화 계층의 런어블까지 포함하면 총 14개의 런어블들이 존재한다. 제어의 흐름에 의해 구분된 런어블들은 OS 스케줄러에 의해 구동될 수 있도록 RTE에서 태스크 바디를 생성해야 한다. 가장 단순하게 설계한다면 런어블 개수와 동일한 태스크 바디를 생성하여 각각 하나씩 태스크와 런어블의 쌍을 만들어서 단순하게 정적 스케줄러로 설계할 수 있다. 하지만 이 방법은 OS의 태스크는 개수가 제한되어 있고 불필요한 태스크 스위칭에 의해 시간 낭비를 초래하기 때문에 적합하지 않다. 런어블의 입출력 인터페이스와 실행순서, 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 6에서 모터 구동과 출력에 관련된 런어블들은 샘플링 주기가 60.0us으로 다른 런어블들에 비해 수십배 이상 차이가 나기 때문에 태스크 바디를 따로 생성하여 할당해야 한다. 그리고 EPS의 실행여부를 결정하는 EPS Enable Manager 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.

### 3. 타이밍 결정과 동기화

맵핑이 끝난 런어블들의 실행순서를 EPS 제어 알고리즘의 실행순서와 동일하게 정의함으로써 실제 제어되는 데이터의 흐름과 OS 스케줄러 제어순서가 일치하여 시간 동기화가 가능하다. 또한 태스크의 최대 주기를 계산하여 런어블의 타이밍 주기를 결정하고 트리거 타이밍을 동기화 시켜주는 것이 필요하다. 트리거 타이밍을 동기화하기 위해서는 먼저 태스크의 최대주기 즉 GCD (Greatest Common Divisor)의 주기를

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 7. 런어블과 태스크의 타이밍 설계.
Fig. 7. Timing design of runnable and task.

분석하는 것이 중요하다. GCD 주기는 런어블의 실행 주기와 오프셋을 통해 계산될 수 있고, 이 때 런어블의 최악실행시간(WCET: Worst Case Execution Time)은 GCD 주기보다 작아야 한다[5,13]. GCD 주기는 맵핑된 모든 런어블들을 활성화하기 위해 결정되는 주기이며 다음과 같이 정의된다.

GCD 주기 = 각 런어블들의 주기와 Activation Offset Time의 최대 공약수

모든 태스크의 GCD 주기를 계산하여 각 태스크의 GCD 주기가 서로 동일하거나 정수 배의 관계로 설계된다면 기본 트리거 주기의 동기화가 되었다고 판단할 수 있다.

섀시 제어 시스템에서는 보통 액추에어터의 제어 주파수가 매우 빠르기 때문에 센서부와 제어부의 샘플링 주기가 최대한 액추에이터의 샘플링 주기에 근접하면서 타이밍이 동기화되는 것이 중요하다. 동기화되지 않는다면 출력 신호의 시간지연, 불규칙한 주기성 등의 문제를 야기할 수 있다.

그림 7은 각 태스크에 맵핑된 런어블들의 실행순서, 태스크의 속성, 그리고 GCD 주기를 계산하여 런어블을 동기화하는 것을 보여준다. 먼저 태스크 2는 EPS의 출력 샘플링 주기가 60.0us으로 일정하게 유지되어야 하고 다른 태스크에 의해 지연이나 중지되어서는 안되기 때문에 높은 우선순위 비선점형 스케줄링 방식을 사용한다. 그리고 태스크 3은 출력 주파수에 영향을 주지 않는 태스크이기 때문에 가장 낮은 우선순위 선점형 스케줄링 방식을 사용한다.

GCD 주기를 계산하여 시간 동기여부를 확인해 보면 그림 8(a)에서 알 수 있듯이, 태스크와 런어블의 최적설계와 시간 동기가 이루어 지지 않은 경우 태스크 간의 시간 동기화가 되지 않고 원치 않는 시간지연이 발생한다. 따라서 시 간 지연이 누적되면 OS 스케줄러의 효율을 떨어뜨리고 제어타이밍이 부정확해 질 수 있다. 이 경우 런어블의 샘플링 주기를 조정하여 GCD를 수정함으로써 (b)와 같이 동기화된 태스크 타이밍 체인을 확인할 수 있다. 또한 태스크 간의 시작 시점이 동일하여 발생할 수 있는 시간지연은 태스크 오프셋을 넣

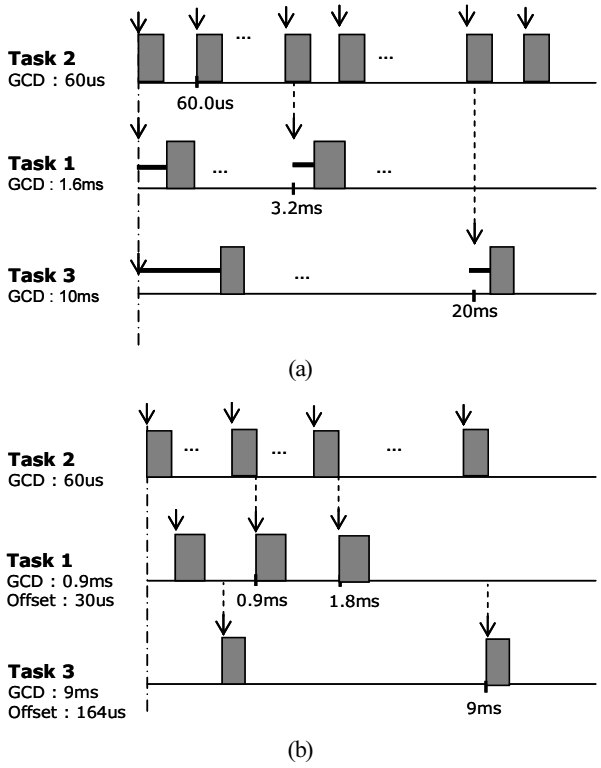



그림 8. 런어블과 태스크의 타이밍 분석.
Fig. 8. Timing analysis of runnable and task.

거나 하나의 태스크로 통합하고 런어블의 실행순서를 조정함으로써 시간 지연의 발생을 막을 수 있다.

## V. 시험 결과

본 장에서는 EPS 스케줄링 설계정보를 기반으로 시뮬레이션을 수행한 결과를 보여준다. 시뮬레이션을 통하여 시간주기, 실행시간 등의 절대적인 수치를 검증하는 것은 불가능하지만 태스크 스케줄링, 타이밍 동기화, 런어블 실행순서 등의 설계 파라미터에 대한 스케줄링 분석과 검증을 하기엔 충분하였다.

### 1. 시뮬레이션 환경

설계된 EPS 소프트웨어의 기능을 검증하고 스케줄링 방법을 적용하여 실제 EPS 시스템의 토크 제어 성능이 개선되는지 평가하기 위하여 그림 9와 같은 시뮬레이션 환경을 구축하였다. AUTOSAR 기반으로 설계된 EPS 소프트웨어 컴포넌트로부터 PC 시뮬레이션을 하기 위한 DLL 파일을 생성하여 SIL (Software In the Loop)기반 시뮬레이션 환경을 제공하는 Vector사의 CANoe 도구와 링크시켰다. CANoe는 RTE 하위의 베이직 소프트웨어를 에뮬레이션하여 가상의 ECU 환경을 제공하기 때문에 베이직 소프트웨어 계층를 제외한 소프트웨어 컴포넌트와 RTE를 쉽게 검증할 수 있다. EPS 시스템에서 모터와 인버터를 포함한 출력 컴포넌트는 고속의 스위칭 주파수로 동작하기 때문에 빠른 실행속도와 Fixed Point 연산이 가능한 Matlab 환경으로 구성하여 실험하였다. 그리고 시뮬레이션 도구 간의 실시간 시간 동기화를 위하여 MATLAB의 Simulink에 있는 CANoe S-Function 블록셋을 사용하여 실시간 동기 시뮬레이션(concurrent-simulation)을 하였다.

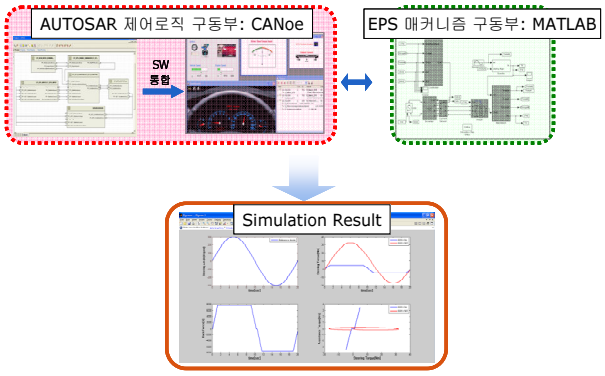



그림 9. EPS 시스템 시뮬레이션 환경.
Fig. 9. Simulation environment for EPS system.

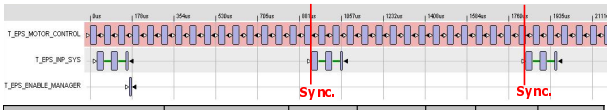
## 2. 타이밍 분석 및 검증

앞서 IV 장에서 언급한 EPS SWC의 스케줄링 설계정보를 이용하여 태스크 기반의 OS 타이밍 분석 도구에서 타이밍 분석과 검증을 수행하였다. EPS 시스템의 출력 샘플링 주기가60us이고 모터구동부의 최악응답시간이 샘플링 주기의 50% 이하가 되어야 한다는 가정 아래 모터 구동 태스크의 데드라인(deadline)을 30us으로 설정하였다. 그 외의 태스크의 실행시간은 런어블 수와 코드라인 등으로 추정한 가상의 값이다. 그리고 Freescale의 OSEKTurbo나 Vector의 osCAN의 경우 태스크 스위칭 시간이 3~4us 정도이기 때문에 본 시뮬레이션에서는 태스크 스위칭 시간을 3us으로 가정하였다. 그 외의 지터 시간과 외부 인터럽트 등은 고려하지 않았다.

그림 10은 스케줄링 최적화와 시간 동기화가 되지 않은 초

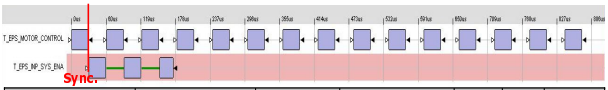

| Task | Priority | Period | WCET (Virtual) | Offset | Time Delay | Synchro -nization |
|---|---|---|---|---|---|---|
| T_EPS_MOTOR_DRIVE | 4 (Non-Preemptive) | 60.0us | 27us | 0 | No | No |
| T_EPS_SYSTEM_CONTROL | 3 (Preemptive) | 3.2ms | 20us | 0 | Yes | No |
| T_EPS_INPUT | 2 (Preemptive) | 1.6ms | 45us | 0 | Yes | No |
| T_EPS_ENABLE_MANAGER | 1 (Preemptive) | 10ms | 10us | 0 | Yes | No |

(a)




| Task | Priority | Period | WCET (Virtual) | Offset | Time Delay | Synchro-nization |
|---|---|---|---|---|---|---|
| T_EPS_MOTOR_DRIVE | 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 |

(b)




| Task | Priority | Period | WCET(Virtual) | Offset | Time Delay | Synchro-nization |
|---|---|---|---|---|---|---|
| T_EPS_MOTOR_DRIVE | 2 (Non-preemptive) | 60.0us | 27us | 0 | No | Yes |
| T_EPS_INP_SYS_ENA | 1 (Preemptive) | 0.9ms | 75us | 30 | No | Yes |

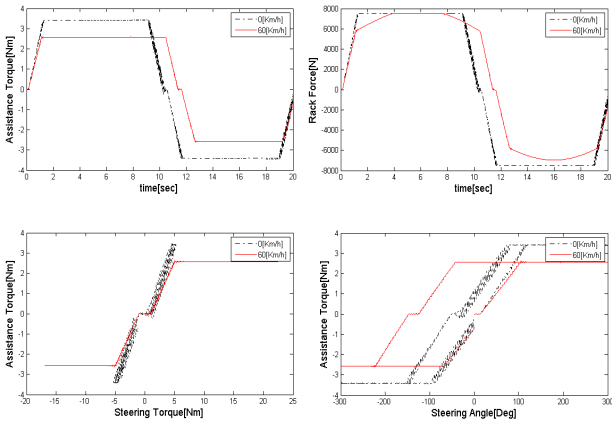
(c)

그림 10. 스케줄링 분석 도구를 사용한 타이밍 검증.
Fig. 10. Timing verification with scheduling analysis tool.

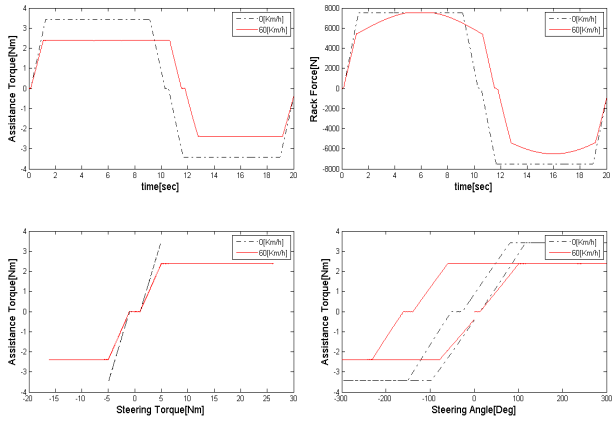
기 설계 상태에서 태스크 동기화 및 최적설계를 하여 EPS 소프트웨어 타이밍 특성을 검증하고 분석하는 과정을 나타낸다. (a)는 스케줄링 최적화와 시간 동기가 이루어지지 않은 초기상태를 나타내고 (b)는 제어흐름과 동일하게 태스크의 우선순위를 수정하고 태스크 수를 줄임으로써 태스크 스위칭 시간을 줄일 수 있도록 최적화 하였다. 또한 태스크 간의 동기화를 맞추기 위하여 오프셋과 주기를 변경하여 전체적으로 개선된 분석 결과를 얻을 수 있었다. (b)에서 보다 빠른 시간 응답성이 요구되면서 태스크 단위의 스케줄링이 필요 없는 경우에는 (c)와 같이 T_EPS_MOTOR_DRIVE 태스크를 제외한 나머지 태스크를 하나의 태스크로 통합하여 태스크 스위칭 시간을 최소화할 수도 있다.

## 3. 시험 결과

그림 11은 그림 10(a)와 (b)에 대하여 시뮬레이션을 한 결과를 나타낸다. 차량 정차 상태와 주행상태(60 Km/h)에 대하여 각각 EPS 시스템의 성능 지표가 될 수 있는 조향 보조토크 곡선, 조향 앵글과 조향 보조 토크 간의 히스테리시스 곡선 등을 비교하였다. 그림 10에서 (a)의 경우에는 잘못된 태스크 스케줄링와 비동기성 등의 영향으로 토크 리플이 심하고 조향 감도가 전체적으로 떨어지는 결과가 나왔고, (b)의 경우에는 비교적 리플이 없고 조향 복귀성과 조향 감도가 개선된 결과를 얻을 수 있었다.




(a)




(b)

그림 11. 시험 결과.
Fig. 11. Test results.

## VI. 결론 및 고찰

오늘날 차량 소프트웨어의 대규모화와 복잡화는 개발비용 증대, 소프트웨어의 신뢰성 및 품질 하락 등을 초래하며 자동차 산업에 심각한 문제로 대두되고 있다. AUTOSAR는 이러한 문제를 해결하기 위해서 설립된 차량 소프트웨어 아키텍처의 표준화 단체이다. AUTOSAR는 표준명세를 통하여 소프트웨어 재사용성과 신뢰성을 향상시키는 등의 긍정적인 효과를 거두었지만, 필연적으로 기존의 소프트웨어보다 시스템에 오버헤드를 증가시킨다. 따라서 제한된 프로세스의 시간과 자원을 효율적으로 사용하기 위해서는 AUTOSAR기반의 태스크 및 런어블의 스케줄링이 적절하게 이루어져야 한다. 특히 프로세스의 요구사양이 높고 빠른 시간 응답성이 요구되는 EPS 섀시 응용시스템에서는 타이밍을 예측하고 스케줄링하는 것이 더욱 중요하다.

본 논문에서는 AUTOSAR기반의 EPS 소프트웨어 컴포넌트의 타이밍 분석 및 최적 스케줄링을 하기 위한 런어블, 태스크의 스케줄링 설계방법을 제시하였다. 그리고 PC기반의 실시간 동기 시뮬레이션 환경을 구축하여 최적 설계된 EPS 소프트웨어 컴포넌트를 검증하였다. 실험결과 런어블 및 태스크의 스케줄링 최적화와 시간 동기화를 함으로써 EPS의 성능 특성이 개선되는 것을 확인할 수 있었다.

시뮬레이션을 통하여 EPS 제어 시스템의 최적 스케줄링 방법의 가능성을 확인하였지만 실제 시스템에 적용하기 위해서는 시간특성 주기, 실행시간 등의 절대적인 수치들을 고려하여 보다 정확한 스케줄링 설계방법이 필요할 것이다.

## 참고문헌


[1] www.AUTOSAR.org.

[2] R. Racu and R. Ernst, “The need of a timing model for the autosar software standard,” *WMAAS 2006*, Dec. 2006.

[3] AUTOSAR “Specification of Operating System,” R3.1, Rev0001, Jan. 2009.

[4] M. Asberg and M. Behnam, “Towards hierarchical scheduling in AUTOSAR,” *ETFA 2009*, Sep. 2009.

[5] AUTOSAR “Specification of RTE,” R3.1, Rev0001, Jun. 2008.

[6] Z. Wu and H. Li, “An improved method of task context switching in OSEK operating system,” *AINA 2006*, Apr. 2006.

[7] G. Liu and A. Kurnia, “A low torque ripple PMSM drive for EPS applications,” *APEC 2004*, Sep. 2004.

[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] B. Jang and J. Jung, “Concurrent simulation of EPS control system and a full vehicle dynamic system,” *SCSC 2004*, Jul. 2004.

[10] AUTOSAR “Explanation of application interfaces of the chassis domain,” R3.1, Rev0001, Jun. 2008.

[11] W. Ramisch and M. Ralf, “TIMMO timing model,” V1.0, Jul. 2009.

[12] A. Ferrari and M. D. Natale, “Time and memory tradeoffs in the implementation of AUTOSAR components,” *EDAA 2009*, Apr. 2009.

[13] O. Scheickl and M. Rudorfer, “How timing interfaces in AUTOSAR can improve distributed development of real-time software,” *Informatik 2008*, Sep. 2008.

[14] AUTOSAR “Specification of timing extensions,” R4.0, Rev0001, Nov. 2009.



**박 광 민**

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



**금 대 현**

2001년 계명대학교 자동차공학과 학사. 2003년 계명대학교 자동차공학과 석사. 현재 경북대학교 전자전기컴퓨터학과 박사과정. 2003년~2005년 LG전자 연구원. 2005년~현재 대구경북과학기술원 연구원. 관심분야는 임베디드 소프트웨어, 테스팅 자동화.



**손 병 점**

2006년 부산대학교 기계공학부 학사. 2010년 부산대학교 기계공학부 석사. 2009년~현재 대구경북과학기술원 연구원. 관심분야는 차량용 네트워크, 시스템 엔지니어링.



**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. 관심분야는 차량 임베디드 시스템.