OSEK OS를 적용한 어플리케이션 효율화

박 원 용, 유 춘 영, 김 재 오, 정 구 민, 문 찬 우, 안 현 식
국민대학교 임베디드 제어시스템 실험실

Efficient Applications Based on OSEK OS

Won-Yong Park, Choon-Young You, Jae-O Kim, Gu-Min Jeong, Chanwoo Moon, and Hyun-Sik Ahn
Embedded Control System Lab., Kookmin University

Abstract - 본 논문에서는 차량용 실시간 운영체제인 OSEK/VDX가 어떠한 구조로 이루어졌는지 알아보고 간단한 Application에 응용해 보았다. OSEK/VDX의 표준화 되어있는 기본 구성요소인 OS, COM, NM, OIL에 대하여 각각의 기능에 대하여 소개하고 OIL파일을 작성해서 ECU에 적용하기 위한 설계 가이드를 제시한다.

1. 서 론

최근의 자동차에서는 연비, 동특성, 안정성 및 안락성을 향상시키기 위하여 다양한 전자제어장치(Electronic Control Unit : ECU)들의 적용이 증가하고 있다. ECU의 적용이 증가함에 따라 ECU의 핵심 역할을 담당하는 MCU(Micro Controller Unit)의 사용역시 증가하고 있다. 2003년 메르세데스 벤츠의 S클래스의 경우 50개 이상의 ECU들이 3개의 네트워크로 연결되어 150여 종류의 메시지를 통해 600여개의 신호를 주고받고 있다 [1].

이러한 자동차 전자제어장치의 증가와 더불어 소프트웨어의 복잡성 또한 증가하고 있으며 높은 수준의 ECU와 S/W를 개발해야한다는 필요성이 대두되고 있다. 그리고 날로 늘어가는 시장의 요구와 짧은 개발주기로 인해 S/W의 표준화의 필요성 또한 중요한 사항으로 인식되기 시작 하였다.

안정적이고 효율적으로 ECU를 관리하고 S/W 개발자의 효율을 향상시키기 위한 방법 중의 하나가 표준화된 실시간 운영체제 시스템을 도입하는 것이다. 유럽과 일본에서는 운영체제에 대한 표준화가 이루어져서 대표적으로는 유럽에서 개발된 OSEK/VDX가 있으며, 표준 소프트웨어 플랫폼으로서 유럽의 AUTOSAR 및 일본의 JASPAR 등이 있다. 국내 완성차 업체와 부품업체도 OSEK/VDX와 AUTOSAR 표준 도입에 적극적이며 BMW에서 표준 OS로 사용된 일렉트로비트사의 'TRESOS'와 같은 OSEK OS 제품을 도입해 유럽 수준의 소프트웨어 플랫폼 구축을 목표로 진행하는 사례가 많아지고 있다 [2].

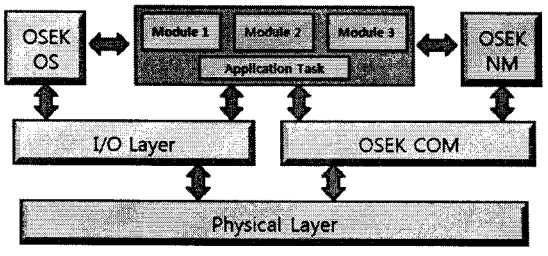
본 논문에서는 OSEK/VDX의 기본 구성요소를 소개하고 일렉트로비트사의 TRESOS를 이용하여 OSEK/VDX를 적용한 간단한 예제 시스템을 설계함 으로써 Task간의 상호관계를 보여주고 실제 전자제어시스템에 OSEK/VDX를 적용할 때의 설계 가이드를 제시 한다.

2. OSEK/VDX 소개

2.1 OSEK/VDX

OSEK은 1993년 독일의 자동차 회사들의 합작 프로젝트로 만들어 졌으며 차량 내 전자시스템의 표준을 겨냥해 설계된 실시간 운영체제 이다. 개발 초창기에는 BMW, Bosch, Daimler Chrysler, Opel, Siemens, VW 그리고 IIIT of the University of Karlsruhe가 개발에 참여 하였으며 1994년에 프랑스의 차량 제조업체인 PSA와 Renault사가 개발한 차량 내 분산시스템(Vehicle Distributed eXecutive : VDX)과 병합하면서 OSEK/VDX로 명명되었다.

OSEK는 독일어로 "Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug"의 약자이며 영어로는 "Open Systems and the Corresponding Interfaces for Automotive Electronics" 이다. 현재 OSEK/VDX는 Operating System(OS), Communication(COM), NM(Network Management), 그리고 OIL(OSEK Implementation language)의 4개의 표준규격을 포함하고 있으며 ORTI(OSEK/VDX Real-Time Interface, OSEK/VDX Time-Triggered Operating System, 그리고 OSEK/VDX Fault Tolerant Communication specification의 3개의 추가적인 표준규격이 진행 중에 있다 [3].

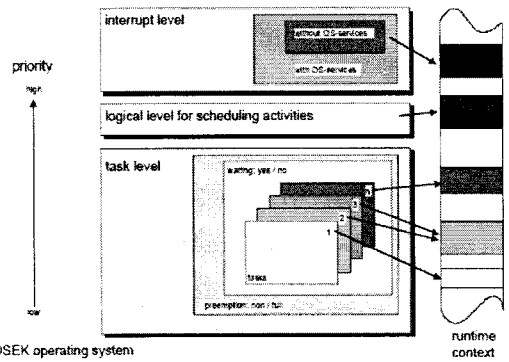


<그림 1> Typical OSEK/VDX Application

<그림 1>은 전형적인 OSEK/VDX를 사용한 어플리케이션의 그림을 나타낸다. OSEK/VDX를 사용하는 가장 큰 목적중의 하나가 코드의 재사용 및 이식성이다. 각각의 모듈을 구성할 때 OSEK/VDX에서 제공하는 표준화된 API를 사용 함 으로써 차후 소프트웨어의 유지보수 및 개발 비용을 줄일 수 있고 다른 프로세서로의 이식 또한 코드의 많은 변경 없이 가능하다. Physical Layer는 MCU로 구성될 수 있으며 I/O Layer는 OSEK/VDX에 표준화되어 포함되어 있지 않고 Physical Layer에 맞춰 개발자가 구성 할 수 있다. OSEK COM과 OSEK NM은 프로세서 및 태스크간 통신을 지원한다.

2.1.1 OSEK Operating System

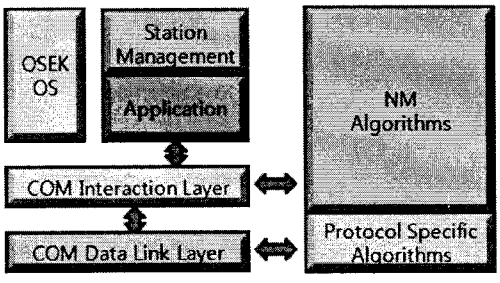
OSEK Operating System은 OSEK/VDX에서 기본적으로 제공하는 실시간 운영체제로써 전체적인 코드의 크기가 작고, 확장에 용이하며 실시간 멀티태스킹을 지원한다는 특징이 있다. 8Kb의 작은 메모리를 가진 8bit MCU에서 512Kb이상의 큰 메모리를 가진 32bit MCU까지 OSEK OS를 적용할 수 있다. OS는 기본적인 실시간 Task관리, Timer를 사용한 Counter, Alarm기능, 인터럽트 처리, 자원 공유 및 관리, Event를 이용한 Task 동기화, 그리고 Task간의 간단한 통신을 지원한다. OSEK OS는 기본적으로 Running, Ready 그리고 Suspended의 세 가지의 Task 상태를 제공하며 상황에 따라 Waiting을 추가한 네 가지의 Task 상태를 사용하도록 설정할 수 있다.



<그림 2> Processing levels of the OSEK operating system

OSEK OS는 세단계로 처리 순서로 나눌 수 있고 <그림 2>는 처리순서의 우선순위를 나타낸다. Task level에서 Task의 우선순위는 개발자에 의해 할당되고, 할당된 우선순위에 따라 Task들이 Scheduling된다. OSEK OS에서는 Scheduling방법으로 선점형, 비선점형 둘 다 제공할 뿐만 아니라 선점형과 비선점형을 섞어서 사용하는 Mixed형태의 Scheduling 방법도 사용할 수 있다. Interrupt level은 모든 Task의 우선순위보다 앞서고 Interrupt 자체의 고유 우선순위에 따라 실행이 결정된다.

2.1.2 OSEK Communication 및 NM



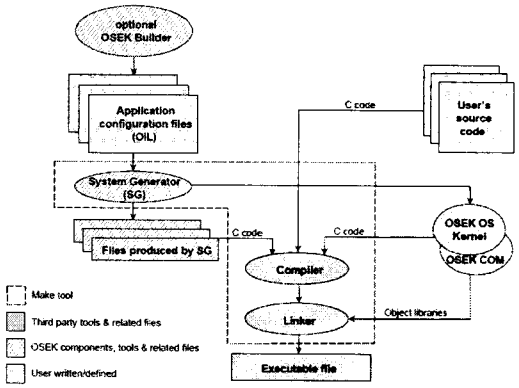
<그림 3> OSEK COM and NM Environment

OSEK/VDX의 두 번째 표준으로 제공되는 OSEK COM은 어플리케이션 내부의 Task 및 Process간의 통신 프로토콜과 인터페이스를 제공한다. <그림 3>은 OSEK NM의 기본 구성요소와 OSEK COM 그리고 어플리케이션과의 상호 관계를 나타낸 그림이다. OSEK COM은 Interaction Layer를 전체적으로 포함하고 Network Layer와 Data Link Layer를 부분적으로 포함한다 [4].

OSEK NM은 네트워크에 연결되어 있는 노드의 네트워크 상태를 모니터링 할 수 있는 기능과 NM구성요소의 제어를 위한 API(Application Program Interface)를 제공한다. OSEK NM은 직접 네트워크 관리와 간접 네트워크 관리 방식의 두 가지 알고리즘으로 구성되어 있으며 제공되는 API는 NM을 제어하는 API와 자신의 네트워크 상태 및 다른 노드의 네트워크 상태를 모니터링 하는 API를 제공한다 [5].

3. OSEK/VDX 기본응용

3.1 어플리케이션 개발과정



<그림 4> Example of development process of application

<그림 4>는 OSEK/VDX를 응용한 간단한 개발과정을 보여준다. 그림5는 ECU가 네트워크에 연결된 상태가 아닌 단일 ECU로써 OSEK/VDX를 사용하여 개발 할 때의 과정을 나타낸다. OIL파일은 특정 어플리케이션에 맞게 OSEK/VDX를 설계 할 때 사용하는 언어이며 사용자가 직접 작성하거나 OIL파일을 생성시켜주는 Tool을 사용하여야 한다. System Generator를 거친 OIL파일은 주로 C언어로 변환되어 OSEK Kernel에 삽입되거나 외부 파일에 C코드로 저장된다 [6].

본 논문에서는 OIL파일과 OSEK Kernel을 생성시키는 System Generator로써 3SOFT사의 TRESOS를 사용하였다. System Generator로부터 만들어진 파일과 OSEK Kernel은 함께 컴파일 되고 링크 되어야 하며 Compiler로는 Altium사의 Tasking을 사용하였고 OSEK/VDX를 적용할 어플리케이션 타겟 보드로는 인피니언사의 XC167CI 마이크로 컨트롤러 보드를 사용하였다.

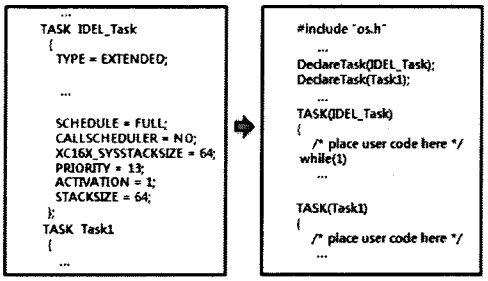
3.1.1 OIL을 사용한 기본 시스템 설정

```
CPU OSEK_XC16X {
OS new_OS {
  ...
  CC = ECC2;
  SCHEDULE = FULL;
  STATUS = EXTENDED;
  ...
  }
COUNTER new_COUNTER {
  ...
  }
ALARM new_ALARM {
  ...
  }
...
};
```

<그림 5> OIL definitions for initial application

<그림 5>는 초기 OIL파일을 통하여 전체 시스템을 설계 할 때의 예제를 나타내었다. 이 예제에서는 단일 MCU를 사용하였기 때문에 CPU Object는 한 개이고 OSEK COM과 OSEK NM은 사용하지 않았다. 내부적으로 OS의 전반적인 설정을 담당하는 OS Object와 Counter, Alarm, Resources, Event, Interrupts, Task 등의 설정을 담당할 각각의 Object 들을 정의 할 수 있다. OIL로부터 설정된 값에 따라 전체시스템의 구조가 설계되고 시스템의 구조가 완성되었다면 각각의 Object세부의 설정 값을 변경하여 시스템 특성을 변경시킬 수 있다.

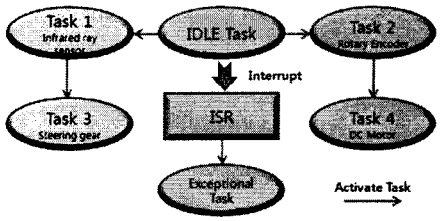
3.1.2 Task 세부설정



<그림 6> Example of transformation

<그림 6>은 OIL 파일내부의 Task관련 설정과 C언어로의 변환을 나타낸다. OSEK/VDX는 최소한 한 개 이상의 Task를 가져야 하며 기본적인 IDLE Task를 제공하지 않으므로 최소 1개의 Task를 사용자가 생성시켜야 한다. OIL 파일 내에 각각의 Task 설정을 담당할 Object들을 정의하고 작성된 정의에 따라 Task는 고유의 우선순위와 Stack크기 등이 설정되며 OIL 파일의 C언어 변환이 끝났다면 각각의 Task 동작을 작성해야 한다.

3.2 응용 예제



<그림 7> Example of basic application

<그림 7>은 여러 개의 Task로 구성된 간단한 Line Tracer 어플리케이션의 동작을 나타낸다. 모든 Task가 Suspended 되어있는 상태에서 IDLE Task는 시스템 시작과 동시에 Activate된다. IDLE Task는 Counter, Alarm 기능을 사용하여 만든 Sampling Time에 따라서 Task1과 Task2를 Activate시켜 주고 Task1과 Task2는 공유하는 자원이 없으므로 우선순위에 따라 실행이 결정 된다. Task1과 Task3 그리고 Task2와 Task4는 서로 종속관계에 있으므로 Event를 통하여 동기화를 시켜주어야 한다. Task1 또는 Task2에서 센서 값에 대한 처리가 끝났다면 하위 Task로 메시지를 보내주고 Task3과 Task4에서는 각각 상위 Task로부터 받은 메시지에 따라 actuator를 구동하게 된다.

4. 결 론

본 논문에서는 차량용 실시간 운영체제인 OSEK/VDX를 소개하고 간단한 Application에 적용해 보았다. 향후 차량 내 전자제어 시스템은 그 수의 증가와 소프트웨어의 복잡성이 더욱 증가함으로써 짧은 시간 내 높은 수준의 소프트웨어 개발과 표준화의 필요성이 점차 증대될 것이다. OSEK/VDX는 표준화된 플랫폼을 제공하며 상이한 MCU환경에도 적용 가능한 확장성을 제공한다. 뿐만 아니라 S/W개발자의 효율을 증대시키기 위한 간단하고 체계적인 API를 제공한다. OSEK/VDX를 적용한 간단한 예제 시스템을 통해 Task간의 상호관계를 보여주었고 OSEK/VDX에서 제공하는 표준화된 API로 예제를 설계함으로써 향후 다른 타겟보드로의 이식이 용이하고 유지보수 및 개발시간 또한 단축될 것으로 보인다.

[참 고 문 헌]

[1] Klaus Grimm, "Software Technology in an Automotive Company - Major Challenges", Proceedings of the 25th International Conference on Software Engineering, pp498-503, IEEE, 2003

[2] 김승룡, "자동차용 운영체제", Digital Times, 2007

[3] OSEK/VDX Operation System Specification 2.2.3, OSEK/VDX steering committee, 2005

[4] OSEK/VDX Communication Specification 3.0.3, OSEK/VDX steering committee, 2005

[5] OSEK/VDX Network Management Specification 2.5.3, OSEK/VDX steering committee, 2005

[6] OSEK/VDX OSEK Implementation Language Specification 2.5, OSEK/VDX steering committee, 2005

[7] "ELEKTROBIT OSEK/VDX OS Fundamentals, Design and Implementation",

[8] JOSEPH LEMIEUX, "Programming in the OSEK/CDX Environment", CMP books, pp.1-256, 2001