차량용 전장소프트웨어 운영체제를 위한 AUTOSAR OS (OSEK)과 상용 Real Time Operating System (RTOS) 비교

심재연, 김성환
서울시립대학교 컴퓨터과학부
e-mail : simpo@uos.ac.kr, swkim7@uos.ac.kr

Comparison Analysis of Candidate Real Time Operating System including AUTOSAR OS (OESK) for Automobile Operation

Jae-Youn Shim and Seong-Whan Kim
Dept. of Computer Science, University of Seoul

요 약

자동차산업이 발달하면서 자동차 내에도 ECU 기반의 전자 장비가 증가하고 있다. 다양한 차량용 전자 장비의 소프트웨어의 호환성과 효율적인 구동을 위하여 독일 자동차 업체를 필두로 자동차 업계에서는 OSEK/VDX 운영체제를 기반으로 AUTOSAR 표준화를 제안하였다. 본 논문에서는 AUTOSAR 표준에서 기반으로 하고 있는 OSEK/VDX 와 기존의 RTOS (실시간 운영체제)를 다양한 측면에서 비교 분석하여 향후 자동차 소프트웨어를 위한 운영체제의 발전방향과 각각의 운영체제의 특성과 장점을 분석 하였다.

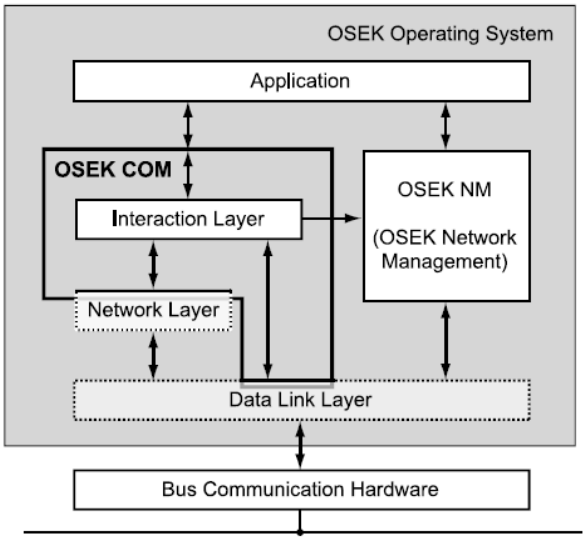
1. 서론

현대 자동차 산업의 전자기술은 단순히 부품을 제어하거나 단순한 기능을 제공하는 것이 아니라 좀더 복잡한 기능을 제공하고 통합적인 시스템 관리를 통해 다양한 서비스를 구현한다. 예를 들어, 자동차 내부에 탑재되는 전자 부품의 개수는 1958 년 라디오를 조작하기 위한 단 개의 장비에서 2010 년 61 개 이상의 전자 장비가 들어간다. 이처럼 자동차 내의 전자 장비는 기하 급수 적으로 늘어나고 이에 따른 소프트웨어의 개발 또한 필요하게 됐다. 각각의 전자 장치가 탑재된 부품들이 늘어나며 서로간의 호환성과 공통 모듈사용에 대한 운영에 있어서 문제가 발생 하였는데 이를 해결하기 위해 표준화된 차량용 소프트웨어 플랫폼 개발이 요구 되었으며, 독일의 자동차 업계를 중심으로 자동차 응용 소프트웨어의 이식성과 재활용성을 지원하는 것을 목표로, 자동차용 운영체제 표준 규격인 OSEK/VDX 와 자동차용 미들웨어 표준 규격인 AUTISAR 오픈 플랫폼을 제안하였다. OSEK 는 독일어로 “Offend Systeme und deren Schnittstellen fur die Elektronik im Kraftfahrzeug” 라는 말의 약자로 영어로는 “Open System and the Corresponding Interfaces for Automotive Electronic”을 의미 한다. 독일의 자동차 업계의 공동 프로젝트로 시작 하였으며, 프랑스 자동차 산업계의 VDX-approach(Vehicle Distributed eXecutive)프로젝트와 결합하여 OSEK/VDX 로 발전 하였다 [1].

본 논문에서는 대표적인 실시간 운영체제와 차량용 표준 운영체제인 OSEK/VDX 를 다양한 측면에서 비교 분석하고자 한다. 본 논문의 2 장에서는 자동차용 운영체제인 OSEK/VDX 의 구성과 특성을 설명하고, 3 장에서는 기존 실시간 운영체제인 VRTX, VxWorks, Real-Time Mach, Real-Time Linux 대하여 설명한다. 4 장에서는 자동차용 전장소프트웨어를 위한 운영체제로 OSEK/VDX 와 기존의 실시간 운영체제를 비교 분석하고, 5 장에서 결론을 제시한다.

2. 자동차 전장소프트웨어를 위한 운영체제

본 장에서는 차량용 운영체제의 표준인 OSEK/VDX [6]을 소개한다.



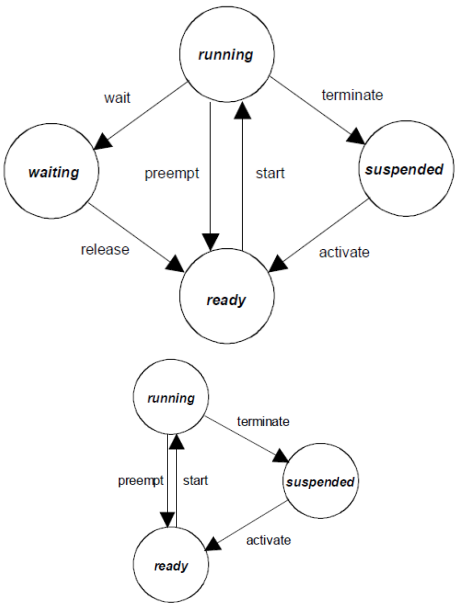
(그림 1) OSEK/VDX 의 전체구조

<그림 1>의 OSEK/VDX 의 구성과 같이, OSEK 표준의

구성요소는 운영체제 구성을 위한 OSEK OS, OSEKtime, 통신스택인 OSEK COM, OSEK FTCom, 네트워크 관리 프로토콜에 대한 표준 인터페이스인 OSEK NM, 사용자가 CPU 및 응용 의 구성 네트워크 구성에 대한 설정을 할 수 있도록 지원하는 OSEK OIL, 디버거를 통해 운영체제 내부 데이터를 쉽게 볼 수 있도록 지원하는 OSEK RTI 로 구성된다.

2.1. OSEK/VDX Multi-Tasking

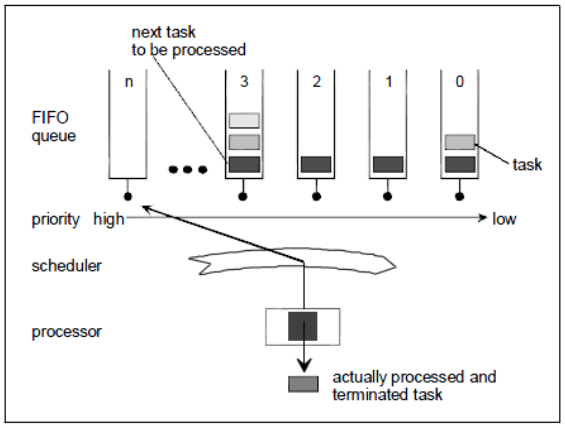
OSEK OS 는 기본적으로 선점형 (preemptive) 멀티태스킹을 지원하는 기본적인 운영체제이며 응용 프로그램에 표준화된 인터페이스를 제공함으로써 HW 에 독립적인 응용 개발을 가능케 하며, 확장성과 안정성을 높일 수 있다. 멀티태스킹을 위한 Task 의 생성, 이벤트 기반의 태스킹 스케줄링, 이벤트/자원 관리, 인터럽트 관리, Alarm, Task 동기화 기능을 제공한다. Task 는 Running, Ready, WAITING (Task 가 어떤 이벤트가 발생하기를 기다리며 수행을 일시 중단한 상태로 이벤트가 발색하면 즉시 Ready 상태가 됨), Suspended 의 네 가지 상태를 가질 수 있다.
OSEKOS 는 Task 를 Extended Task 와 Basic Task 로 구분함으로 제한된 시스템 자원을 효율적으로 사용한다. <그림 2>에서 보는 바와 같이, Basic Task 는 Extended Task 의 상태와 달리 WAITING 상태를 지원하지 않는다. 즉, Basic Task 는 CPU 를 다른 Task 로 넘겨주지 않고 실행을 완료하는 Task 로 Task 가 활성화 될 경우 대기 큐에 등록되어 멀티태스킹을 지원한다. Extended Task 는 Task 가 선점되는 경우를 포함하여, 이벤트를 사용하여 더 낮은 우선순위를 가지는 Task 에게 CPU 를 넘겨줄 수 있는 Task 이다.



(그림 2) Task 상태 변화: (a) Extended, (b) Basic Task

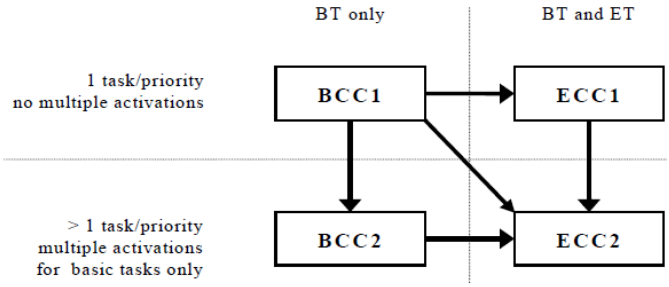
2.2. OSEK/VDX Task Scheduling

OSEK 은 <그림 3>과 같이, 개별 task 에 정적으로 우선순위를 할당하고, 우선순위를 기반으로 한 스케줄링 방식을 채택함으로 기본적으로 선점형 멀티 태스킹을 지원하지만, 선점형 스케줄링 방식, 비선점형 스케줄링 방식, 혼합형 스케줄링 방식을 제공한다.



(그림 3) 스케줄러 동작

OSEK OS 는 Conformance Class (CC)라 불리는 4 개의 OS 프로파일을 가진다. BCC1, BCC2, ECC1, ECC2 로 나누어지며 어플리케이션의 특성에 따라 필요한 기능만 구현된 OS 를 사용함으로 불필요한 자원 낭비를 막기 위함이다. BCC 와 ECC 로 구분 하는 기준은 Basic Task 만을 다룰 수 있느냐 Basic Task 와 Extended Task 모두를 다룰 수 있는지에 대해 구분되고, CC1 과 CC2 는 여러 개의 Task 가 동일한 우선순위를 가질 수 있는 지에 대한 우선 순위 할당 방식과 Basic Task 의 다중 활성화 가능 회수 에 따라 구분 된다. <그림 4> 는 CC 분류와 하위 호환성 관계를 보여준다. 하위 호환성이란 하위 CC 의 모든 기능을 상위 CC 가 지원하는지 여부이다.



(그림 4) Conformance Class 분류

3. 대표적인 실시간 운영체제 개요

본 장에서는 대표적인 실시간 운영체제를 소개한다.

3.1. VRTX

VRTX (Versatile Real-Time Executive) 는 가장 널리 쓰이는 실시간 OS 의 한가지로 멘토 그래픽사에서 개발하였으며 극히 제약된 리소스를 가진 간단한 시스템을 위한 VRTXoc(On-Chip)와 메모리 보호가 필요한 복잡한 시스템을 위한 VRTXsa(Scalable Architecture) 두 종류의 커널을 가진다[2]. 네트워크를 통한 debugging 작업이 가능하며, C 코드에 의한 source level debugger 도 지원한다. VRTX 는 실시간 OS 의 기

본 기능인 Task 의 실시간 처리 및 멀티태스킹 기능을 모두 지원하며 기존의 OS 에 네트워크 기능을 추가하였다.

3.2. VxWorks

VxWorks 는 WindRiver Systems 에서 개발한 실시간 운영체제로 선점형 스케줄러 기반의 빠른 멀티태스킹 커널로 빠른 인터럽트 응답 시간과 확장된 ITC(Inter Task Communication)을 지원 한다. 0 부터 255 의 총 256 단계 우선순위를 사용하며 동일한 우선순위의 Task 사이에는 선택적 라운드로빈 스케줄러를 사용하고 POSIX 인터페이스와 AMP(Asymmetric Multi-Processor)를 지원한다[3]. 또한 VxWorks 는 사용자 인터페이스를 위한 WindSh 라 부르는 Shell 을 지원하고 Shell 은 심볼릭 또는 소스 수준의 디버깅 기능과 성능 모니터와 파일 시스템의 입출력 등을 지원한다. VxWorks 커널의 대표적인 기능은 multitasking, 우선순위 스케줄기능, Task 간의 동기 및 Task 간의 데이터 전송, 인터럽트 처리 지원, watchdog 타이머, 메모리 관리 등의 기능을 지원한다. 또한 Unix 와 BSD Unix 호환 소켓을 통하여 원격 명령처리, 원격 로그인, 원격 함수 호출 (RPC) 등과, ARP, BOOTP 등의 인터넷의 표준 프로토콜을 모두 지원하며, Object 코드를 자체 디스크 또는 네트워크를 통하여 run-time relocation 과 링크작업을 하면서 dynamic loading 과 unloading 을 수행한다. VxWorks 의 함수들과 응용프로그램의 함수를 C 에 의하여 연결할 수 있으며 소스 레벨, C-서브루틴, disassembler, break-point, 싱글 스텝, 시스템 상태표시, 어드레스 및 버스 오류 등을 포함한 각종 인터럽트를 처리할 수 있다. Unix 및 Win95 등의 입, 출력 시스템과 호환하여 지원되며 파일 시스템을 지원한다. 실시간 OS 의 성능을 최적화 할 수 있도록 시스템의 실행함수 및 서브루틴들 실행시간을 측정할 수 있도록 각 함수 및 서브루틴에 타이머를 지원한다.

3.3. Real-Time Mach

Real-Time Mach(RT–Mach)는 Carnegie Mellon University 의 Advanced Real-Time Technology 에서 개발 한 것으로 RT-Mach 의 목적은 사용자에게 예측가능하고 신뢰성 있는 분산 실시간 환경을 제공하는 Mach 커널의 실시간 버전으로 개발 되었다[4]. Mach 마이크로 커널은 IPC, 단기 스케줄링 메모리 관리 같은 필수적인 서비스만 갖는 운영체제이다. RT-Mach 의 특징은 스레드 모델에 시간 속성을 추가한 실시간 스레드 모델, 통합된 실시간 스케줄러의 제공, 하드웨어 자원을 관리하기 위한 수단, 다양한 실시간 동기화 메카니즘, 메모리 상주 객체 지원 기능을 제공한다.

3.4. Real-Time Linux

Real-Time Linux(RT-Linux)는 Linux 의 드라이버 관리 X 윈도우 시스템, 네트워크와 같은 구성요소를 그대로 가지면서 Linux 를 실시간 운영체제로 전환시킨 코드이다[5]. Linux 커널 과 하드웨어 사이에 실시간 executive 를 넣는다는 단순한 아이디어에서 출발하고 있으며, Executive 는 기본적인 커널 수준에서 실시간 Task 의 수행을 지원하며 커널 위의 Linux 프로세스 Task 중 하나로 간주 하고 수행한다. 실시간 애플리케이션은 실시간 Task 들의 모임으로 정의하고 Linux 프로세스는 수행을 원하는 Task 가 없을 경우 가장 낮은 우선순위를 갖는 Task 가 된다. 이를 통해 RT-Linux 는 실시간 Task 와 Linux 프로세스 두 가지 독립적인 계층을 가진다. 실시간 Task 들은 커널의 특권을 가지고 커널 수중에서 수행되며 Linux 와 모든 사용자 프로세스는 더 이상 수행할 실시간 Task 가 없을 때 수행된다. 또한 Linux 프로세스는 사용자 수준과 커널 수준에 상관없이 보다 높은 우선순위의 Task 가 활성화 될 때 마다 선점 당한다. 예측 할 수 없는 인터럽트 디스패치 지연을 제거하기 위해 RT-Linux 는 매크로를 애뮬레이트 함으로 원래 함수를 인터럽트 가능 혹은 불가능 하도록 변경 한다. 이러한 매카니즘을 소프트 인터럽트하며 이것은 Linux 커널이 인터럽트를 불가능하게 할 때조차도 실시간 Task 에는 여전히 유용하게 인터럽트를 허용한다.

4. 자동차 전장소프트웨어용 실시간 운영체제 비교

본 장에서는 AUTOSAR OS 인 OSEK/VDX 와 VRTX, VxWorks, Real-Time Mach, Real-Time Linux 를 실시간 특성 측면에서 비교한다.

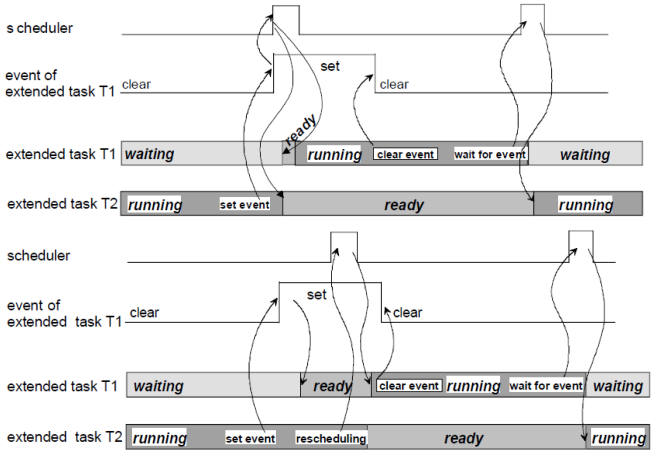
4.1. 멀티태스킹 과 실시간 스케쥴링 비교

대표적인 정적 실시간 스케쥴링 알고리즘으로는, RMS (Rate-Monotonic scheduling)으로 주기를 기초로 Task 의 정적 우선 순위를 할당하고 짧은 주기의 Task 에 높은 순위를 우선적으로 할당하며 Task 가 도착할 때 우선 순위가 할당 되므로 우선순위의 재계산이 필요가 없다. EDF (Earliest-Deadline First scheduling)는 RMS 와 달리 동적인 선점 알고리즘 이다. Task 의 우선 순위가 시간에 따라 변하며 Task 의 종료시간이 짧을수록 우선순위는 높아진다. Round-Robin scheduling 은 선점형 스케줄링의 하나로 시간 단위로 CPU 를 할당하는 방식으로 준비 상태 큐에 진입한 순서대로 처리기를 할당하지만 처리기의 할당 시간을 정의한 후 할당 시간을 초과한 프로세스의 작업은 준비 상태 큐의 끝으로 선점시킨다.

VRTX, VxWorks, Real-Time Mach, Real-Time Linux 모두 FIFO 를 지원 한다. VRTX 는 Preemptive priority-based scheduling 을 사용한다. VxWorks 는 Preemptive Round-Robin scheduling 을 사용한다. Real-Time Linux 는 RMS 와 EDF 을 지원 한다. RT-Mach 는 RMS 기반의 Integrated Time-Driven scheduling 을 지원하며, RMS, Round-Robin scheduling, Deadline Monotonic scheduling, EDF 가 가능하다. OSEK OS 는 스케줄링 정책은 정적으로 할당된 Task 를 우선순위를 기반으로 한 Full preemptive scheduling 방식을 기본으로 하지만 Task 타입에 따라 Non preemptive scheduling 과 Mixed preemptive scheduling 또한 가능하다.

4.2. 이벤트 관리 비교

OSEK OS 의 Event Mechanism 은 동기화 기능을 제공하며, Extended Task 에 적용 된다. 이벤트가 전달되면 우선순위가 높은 WAITING 상태의 Task 는 READY 상태로 전환 된다. <그림 5>는 Full preemptive 방식과 Non preemptive 방식에 대한 Extended Task 의 이벤트 관리에 대해 보여준다. T1 은 가장 높은 순위를 가진 Extended Task 를 나타낸다. 기존의 RTOS 는 이벤트와 함께 데이터 전송 처리가 가능하다. 이벤트의 OVERRUN 신호가 감지 되면 인터럽트 서비스 루틴에서 WAITING 상태의 Task 에 시그널 이벤트를 직접적 보내게 된다.



(그림 5) Event Mechanism: (a) Full preemptive (b) Non-preemptive

4.3 애플리케이션 개발환경 비교

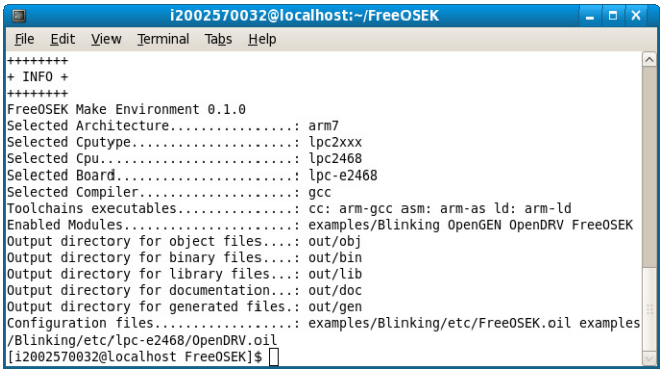
VRTX 와 VxWorks 는 실시간 운영체제의 크기와 성능에 최적화된 커널을 사용하며 상용 운영체제에 비해 주가적인 서비스와 소프트웨어 개발 환경이 상대적으로 약하다. Real-Time Mach, Real-Time Linux 는 상용 운영체제에서 실시간 지원가능을 추가한 것으로 소프트웨어 개발환경이 좋지만 성능과 시간적 정확성이 전용 실시간 운영체제에 비해 상대적으로 약하다.

4.4. 메모리 효율성 비교

OSEK OS 의 Basic Task 는 Task 종료 시까지 실행되며, Basic Task 들은 스택을 공유하여 메모리 절약이 가능하게 사용할 수 있다. 즉, 비선점형 스케줄링 방식의 Basic Task 들은 서로 우선순위가 다르더라도 순차적으로 실행되며, 스택을 공유 할 수 있다. OSEK 의 경우, 불필요한 자원낭비를 막기 위해 CC 를 사용 하는데 항상 모든 기능이 구현되어 있는 ECC2 를 사용하는 대신 어플리케이션을 구성하는 Task 의 특성에 따라 알맞은 CC 를 선택하여 메모리 사용을 최소화 할 수 있다. VRTX, VxWorks, Real-Time Mach, Real-Time Linux 와 같은 기존의 RTOS 의 경우에는 메모리관리를 사용자가 직접 제어 가능하게 함으로써 메모리 효율성을 프로그래머가 관리하게 된다.

4.5. 실험 비교

기존의 실시간 운영체제와 OSEK 비교 실험을 위하여 FreeOSEK 을 사용 하였다[9]. FreeOSEK 는 OSEK/VDX 표준에 의거한 오픈 소스로 POSIX 아키텍처에서 FreeOSEK 을 생성 하기 위해 서는 PHP 5.2.1 이 필요하며 컴파일 하기 위해 gcc 4.1.2 가 필요하다. <그림 6>은 FreeOSEK 제작 환경의 정보이다.



(그림 6) FreeOSEK

5. 결론

본 논문에서 기존의 실시간 운영체제와 OSEKOS 에 대하여 분석하여 보았다. OSEK OS 는 자동차의 다양한 부품에 적용 될 수 있도록 기존의 RTOS 에 비해 효율적인 메모리의 사용이 가능하다.

6. 감사의 글

This work was supported by Business for International Cooperative R&D between Industry, Academy, and Research Institute funded Korea Small and Medium Business Administration in 2010.

참고문헌

[1] OSEK/VDX Binding specification Version 1.4.2, July. 2004
[2] J. Ready, "VRTX: A real-time operating system for embedded microprocessor applications," IEEE Micro, pp. 8-17, Aug. 1986.
[3] 임재석, 손재열, 이용태, 이철훈, "실시간 운영체제 VxWorks 상에서 통신 미들웨어 TAO 의 실시간성 지원에 대한 연구", 한국콘텐츠학회 2009 춘계 종합학술대회 논문집 제 7 권 제 1 호(하), 2009.5, pp: 635-1258
[4] H. Tokuda, T. Nakajima, and P. Rao, “Real-time Mach: Towards a predictable real-time system,” in Proc. Usenix Mach Workshop, Oct. 1990.
[5] Michael Barabanov. "A linux-based real-time operating system." Master's thesis, New Mexico Institute of Mining and Technology, June 1997.
[6] OSEK/VDX OS Version 2.2.3, February 2005
[7] OSEK/VDX COM Version 3.0.3, July. 2004
[8] OSEK/VDX NM Version 2.5.3, July. 2004
[9] http://opensek.sourceforge.net/