OSEK/VDX이용한 실시간 OS 구조에 관한 연구

오창연, 장경식*

*한국기술교육대학교

Study of real-time OS structure that use OSEK/VDX

Chang-Yeon Oh*, Kyung-Sik Jang*

*Korea University of Technology and Education

e-mail : cy75.oh@Samsung.com, ksjang@kut.ac.kr

요약

지금까지 자동차에서의 기술 개발이 엔진 성능과 같은 기계 중점적으로 이루어졌다면 최근에는 전자제어장치(ECU, Electronic Control Unit)를 활용하여 안전, 편의성, 비용 등을 개선하는 전자적 측면의 기술 개발이 활발하게 진행이 되고 있다. 또한, 자동차에 대한 소비자들의 선호가 빠른 속도로 변함에 따라 모델수가 다양해지고 첨단 서비스 개발로 인해 요구되는 소프트웨어의 복잡도가 크게 증가되었다. 이 같은 변화로 자동차 제조비용에서 소프트웨어 개발비용이 상당한 비중을 차지하게 되었고 자동차 업계는 이를 대응하기 위하여 응용 소프트웨어를 모듈별로 재사용할 수 있고 다른 제어장치에도 쉽게 이식할 수 있도록 자동차용 임베디드 시스템의 표준인 OSEK/VDX를 제정하였다.[8] 본고에서는 위의에서 언급한 급변하는 환경을 대응하기 위한 대안으로 제시된 OSEK/VDX표준을 살펴보고, 실제 OSEK/VDX를 이용한 실시간 OS구조에 대하여 알아보자 한다.

ABSTRACT

Technical development in car has utilized electronic controls (ECU, Electronic Control Unit) recently if was achieved machine such as engine performance in priority and electronic side technical development that improve safety, convenience, expense etc. is proceeded vigorously. Also, preference of consumers for car is various model's number according to change at the fast speed and complexity of software required from vanguard service development was increased greatly. Software development expense dominated considerable weight in car manufacture expense by such change and automakers established OSEK/VDX that is standard of automobile embeded system to reuse application software by module to respond this and transplant easily to other control device. Do when search about all item that is necessary in real-time OS structure that examines OSEK/VDX standard that is presented as the alternative to respond environment that change rapidly that refer in dignified mien after it is original, and uses actuality OSEK/VDX.

키워드

Operating system(OS), OSEK/VDX, OIL, ECU

Ⅰ. 서 론

OSEK/VDX는 독일의 Karlsruhe대학과 유럽 자동차 제작회사 그리고 부품 공급업체가 협력하여 발전시킨 Distribute, Real-Time Architecture를 위한 표준입니다. 비록 자동차 산업으로부터 발전되었지만 단지 자동차를 위한 Real Time OS가 아닙니다. 그러므로 OSEK/VDX 표준을 기반으로 한 시스템은 그 외에 최소 자원으로 운영되면서 Compact, Distribute, Real-Time 이 필요로 하는 곳에 사용이 가능합니다. 또한, 다른 많은 OS와는 달리 Open Standard로 다른 프로세서들에 가장

적합하고 효율적인 서로 다른 Implementation을 적용할 수 있습니다. 이점에서 Open Source System인 Linux와 차별됩니다. 즉, Linux 경우 서로 다른 프로세서이지만 Kernel에는 같은 오직 하나의 Implementation만이 존재하게 되며 Implementation 소스는 Open되어 있지만 실질적인 소스제어는 Linus Torvalds' team으로 부터만 이루어지는 제약을 가지고 있는 반면 OSEK/VDX 경우 각각 다른 프로세서들에 가장 적합한 Implementation을 각각 적용할 수 있으며 Open Standard 표준에 준한다면 각각 단체나 개인이 특화된 Implementation 소스를 생성 및 제어를 할 수 있다.

이러한 특징을 가지고 있는 OSEK/VDX는 아래와 같이 4개의 주 표준과 3개의 추가적 표준을 제시하고 있으며

[Main Standard]
(1) The Operating System (OS)
(2) Communication (COM)
(3) Network management (NM)
(4) OSEK Implementation language (OIL)

[Sub Standard]
(1) The OSEK/VDX real-time interface (ORTI)
(2) The OSEK/VDX time triggered operating system
(3) The OSEK/VDX Fault Tolerant communication specification

본고에서는 주 표준인 (1) Operating System (OS), (2) Communication(COM), (3)Network management(NM), (4)OSEK Implementation language(OIL)에 대해서 살펴보고자 한다.

II. 관련 연구

2.1 Operating System (OS)

OSEK OS는 OSEK 의 가장 핵심적인 모듈로 기본적인 실시간 운영체제를 정의한다. 구체적으로 (ㄱ) 멀티태스킹, (ㄴ) 이벤트/자원관리, (ㄷ) 인터럽트 관리, (ㄹ) 알람(alarm) 서비스를 제공한다. 이에 대한 상세내용을 살펴보기로 한다.

2.1.1 멀티태스킹(Task Management)

OSEK OS는 멀티태스킹(ㄱ)을 지원하는 OS로 여러 Task들이 하나의 CPU를 사용 할 수 있도록 Task 상태를 관리하고 Task 실행 순서를 조정하는 Scheduler기능을 가지고 있다.

OSEK OS는 Task 상태관리를 위해 Basic Task와 Extend Task 두 가지 Framework을 제공합니다.

Basic Task와 Extend Task의 큰 차이는 Task 상태관리의 차이로 Basic Task는 3가지 상태(ready state, running state, suspended state)를 관리하는 반면 Extend Task는 Basic Task상태관리에 한 개의 상태가 추가된(waiting state) 4가지 상태를 관리한다. [2][7]

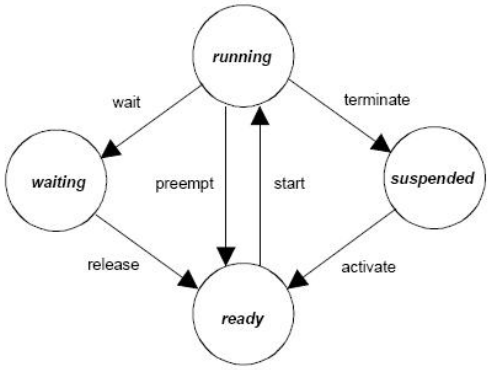

그림1. Extend Task State Model of OSEK OS

각각의 상태에 대한 내용은 다음과 같다.

- Ready: Task가 실행되기 위해 CPU 할당을 기다림
- Running: Task가 CPU을 할당 받아 실행되는 상태
- Waiting: Task가 특정 Event를 기다리기 위해 일시 중지된 상태
- Suspended: Task가 종료된 상태

또한, OSEK OS는 선점형 실시간 운영 체제로 한 Task가 수행중이더라도, 언제든지 필요에 따라 다른 Task가 현재 수행중인 Task를 선점하고 즉시 수행할 수 있다. 선점의 기준은 우선순위로 결정되며 우선순위가 높은 Task가 낮은 우선순위 Task를 선점을 한다. [2][7]

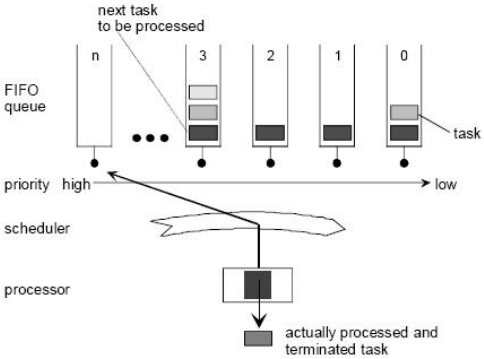

그림2. Scheduler

Scheduler는 다음 수행 할 Task를 결정하는 역할을 담당한다. 그림2 에서 보듯이 Scheduler에는 우선순위 FIFO Queue가 여러 개가 존재하며 Scheduler는 우선순위가 높은 Queue의 Task를 다음 실행 Task로 결정을 하게 된다. 만일 같은

우선순위의 Task가 여러 개 존재 할 경우 가장 오래전에 Ready상태가 된 Task를 선택한다. [2]

다음은 위에서 설명한 Scheduler가 실행 Task를 결정하는 3단계이다.

- Scheduler는 Ready/Running 상태의 모든 Task을 탐색한다.
- Scheduler는 Ready/Running 상태의 모든 Task 중 우선권이 가장 높은 Task 그룹을 선정 한다.
- Scheduler는 우선순위가 높은 그룹 중 가장 오래된 Task를 선정한다.

2.1.2 이벤트/자원관리

이벤트 관리란 Task가 특정 사건을 기다릴 수 있도록 하는 것을 뜻한다. OSEK OS 에서 이벤트는 시스템에서 발생할 수 있는 특정 사건을 나타내는 자료 구조로서 동기화 수단이며 Extend Task에서만 사용할 수 있다. 즉, Task가 특정 이벤트를 기다리기 위해 Waiting 상태로 변경하여 더 이상 수행하지 않거나 특정 Task가 이벤트를 발생 시켜 Waiting 상태의 Task를 Ready상태로 깨어나서 수행을 다시 시작할 수 있게 할 수 있다.

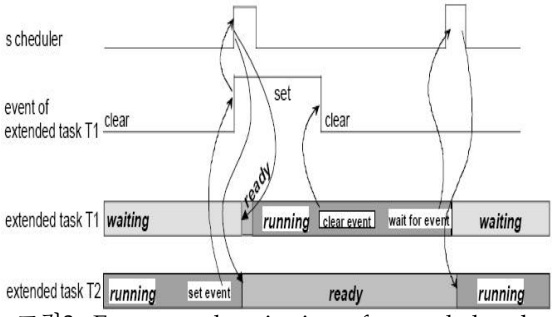

그림3. Event synchronisation of extended tasks

그림3은 Task T1과 Task T2의 동기화 과정에 대한 Time Chart로 Task T2가 이벤트를 발생시키면 이벤트 메시지는 Task T1에 전달이 되고 다시 Scheduler에 의뢰하여 Waiting 상태의 Task T1을 Ready상태를 거쳐 Running 상태로 변경 시키고 Running 상태인 Task T2를 Ready상태로 변경되는 과정과 이벤트 처리가 끝난 다음 다시 원래의 상태로 변경되는 일련의 처리과정을 나타낸 것이다. [2]

자원관리란 태스크들이 자원(메모리, CPU,디바이스등)을 안전하게 공유할 수 있도록 자원의 사용 순서를 조정해주는 것을 뜻한다. OSEK OS 는 데드락과 우선순위 역전현상을 피하기 위해 우선순위 상한 프로토콜 (priority ceiling protocol) 방법을 채택하여 사용하고 있다. 우선순위 상한 프로토콜 방식은 각각의 자원에 Ceiling-Priority를 할당하도록 하게하여 해당 자원을 필요로 하는 Task 중에 우선권이 가장 높은 Task가 Ceiling-Priority를 설정 할 수 있게 해서 한 개의 Task만이 자원을 점유할 수 있도록 하는 방식이다.

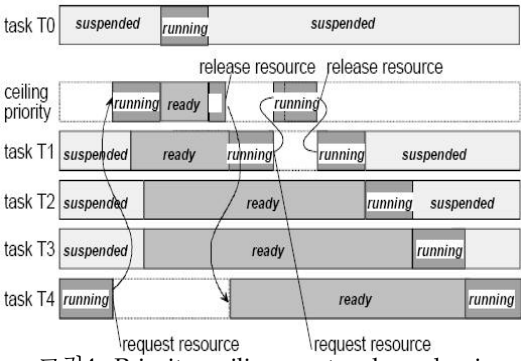

그림4. Priority ceiling protocol mechanism

그림4는 Ceiling-Priority 기법을 설명합니다. 직무 T0는 가장 높은 것, 그리고 직무 T4을 가장 낮은 우선권으로 시킵니다. 직무 T1와 직무 T4는 같은 자원에 접근하기를 원합니다. 우선순위가 높은 Task T1은 Task T4가 자원을 점유한 최대 점유시간보다 짧은 대기시간으로 자원을 점유합니다. [2]

2.1.3 인터럽트관리

많은 Embedded application에서 Interrupt는 외부 사건과 동기화하는데 중요한 역할을 한다. Embedded System에서 사용하는 Interrupt 예로는 Real-Time Clock, Pulse의 흐름을 전송하는 Sensors 그리고 사용 유저가 의도하여 발생시키는 Interrupt이다. [2]

OSEK OS에서는 Interrupt(ㄷ)를 3가지 Category로 나누며 그에 대한 내용은 다음과 같다.

- ISR category 1
  -. 가장 빠른 Interrupt Routine
  -. API 서비스로 Interrupt 요청되지 않음
  -. ISR category 중 최상위에 위치 함

- ISR category 2
  -. API 서비스로 Interrupt 요청
  -. 이 경우 일반적으로 Counter, Task, Event 또는 Message가 필요

- ISR category 3
  -. ISR category 1과 2의 혼합형
  -. 주로는 API함수를 사용하지 않고 Interrupt를 발생시키지만 아주 간혹 API 서비스를 사용
  -. API 서비스 접근은 두 개의 추가된 API서비스로만 접근 가능한 블록 내에서만

접근됨

2.1.4 알람(Alarms)

OSEK OS는 다른 Real-Time OS와는 다르게 Timer개념이 없는 대신 Alarm이 Timer의 기능까지 담당을 하고 있으며, 반복적인 이벤트를 위한 서비스를 제공하기 위해 OSEK OS는 두 단계 개념을 제시하고 있다. [2]

첫 번째는 반복적인 이벤트를 위해 특별한 카운터를 구현하여 레지스터에 저장 시키는 것이고, 두 번째는 카운터 기반의 OSEK OS Application에 Alarm 메커니즘을 반영하는 방법이다.

OSEK/VDX표준은 현재 카운터를 위한 API정의가 없고 Alarm에 대한 API만 정의 되어 있어 결국 카운터 API는 OS를 구현하는 벤더에 따라 달라질 수 있다.

2.2 Communication (COM)

OSEK COM은 태스크들이 서로 통신할 수 있는 방법을 제공해 주는 모듈이다. OSEK COM 의 가장 큰 두 특징은 메시지를 전송단위로 하는 publish/subscribe 통신 모델을 사용한다. [4][7]

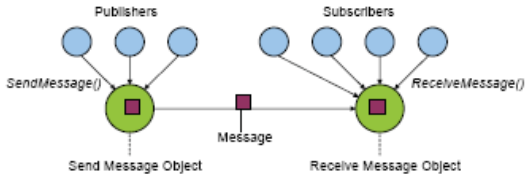

그림5. Communication model of OSEK COM

여기에서는 이 두 특징을 설명하고자 한다.

첫째, 이 모델은 데이터 전송 단위로 메시지를 사용한다. OSEK COM 에서 한 메시지는 메시지의 종류와 메시지의 내용으로 이루어진다. 사용자는 엔진 온도, 타이어 압력, 버튼의 눌림 여부 등 다양한 종류의 메시지를 정의할 수 있다. 메시지의 길이는 비트 단위로 세밀하게 명시할 수 있다.

둘째, 메시지는 publish/subscribe 방식으로 전송된다. 그림 5에서 볼 수 있듯이, 이 방식을 사용하면 메시지를 보내고자 하는 태스크(publisher)는 통신 서브시스템에 메시지를 전달(SendMessage())하기만 하면 되고, 메시지를 받고자 하는 태스크(subscriber)는 통신 서브시스템으로부터 메시지를 구독(ReceiveMessage())하기만 된다. Publisher 가 특정 종류의 메시지를 OSEK COM 에게 전달하면, OSEK COM 은 그 종류의 메시지를 요청하는 모든 subscriber 들에게 메시지를 전송하는 역할을 한다. 이렇게 publisher 와 subscriber 가 서로 느슨하게 연결되어 있기 때문에, 태스크들은 서로 통신하기 위해 상대방의 존재를 알거나, 시스템의 전체적인 연결 상태를 파악할 필요가 없다는 장점이 있다. 또한 새로운 태스크를 추가하는 것이 매우 쉬우며, broadcast 를 전송메커니즘으로 하는 자동차용 네트워크에 매우 효과적으로 구현이 가능하다.

2.3 Network management (NM)

OSEK NM은 Node 모니터링 서비스를 정의한다. 구체적으로 Direct Network Management와 Indirect Network Management 에 방법에 대하여 설명한다. [3]

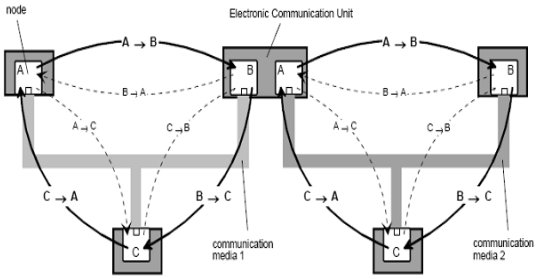

그림6. Infrastructure of NM(logical ling)

Direct Network Management에서의 Node 모니터링 서비스는 Network상의 모든 Node들을 직접 모니터링 하는 방식으로 Network 전반에 동기화가 요구되어 논리적 링 방식을 사용한다. 그러므로 각각의 Node는 논리적 Successor가 할당되고 논리적 첫 번째 Node는 논리적 마지막 Node의 Successor가 된다. 이렇게 하여 전체 상당한 양의 Message들의 분산 제어를 보장 할 수 있게 된다. 그림6은 링의 논리적 연결 모습을 보여주고 있다.

Indirect Network Management는 System 양상에 따라 Node들의 직접적인 모니터링이 불가 또는 바람직하지 않을 수가 있어 Indirect 모니터링 방식이 소개되었다. 이 관리 방식은 모니터된 응용메세지의 용도를 기반으로 모니터하는 방식으로 정상적으로 동작하는 Node로 제한된다. Node의 상태와 Network 상태를 평가 하기위해 Transmission, Reception, Status signal 3가지 비베타적 방법을 제공한다.

- Transmission
  - Transmission 모니터링 스키마를 사용한 전송상태 체크
- Reception
  - Reception 모니터링 스키마를 사용한 수신상태 체크
- Status signa
  - Data link layer의 사용 제어기에 의한 Network 인터페이스 상태 체크

2.4 OSEK Implementation language (OIL)

OIL은 OSEK OS, OSEK COM 등 다양한 OSEK모듈들과 응용 프로그램의 속성을 설정하는 규약이다. 예를 들어 사용자는 OIL 을 작성하여, ECU 에 어떠한 OSEK 모듈들이 포함될 지, 어떤 Task 들이 수행되며, 어떤 이벤트, 자원, 메시지들이 사용될 지 등을 정할 수 있다. 또한 각 태스크의 우선순위나 메시지의 크기, 사용될 통신 옵션의 종류 등을 정할 수도 있다. [7]

III. 결 론

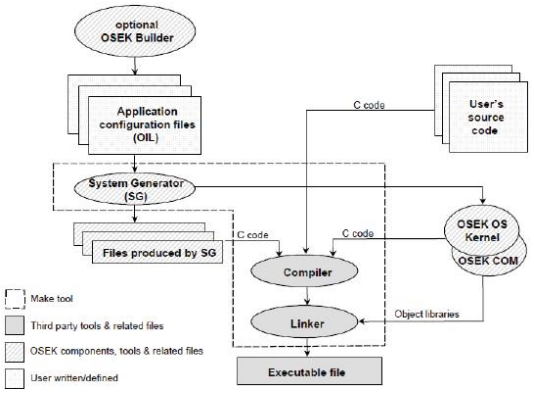

그림7. Development process of OSEK App.

여기까지 최근 자동차에 탑재하는 실시간 소프트웨어 플랫폼인 OSEK/VDX 구조에 대해 알아보았다. 끝으로 OSEK OS 이용한 Application개발 과정을 살펴봄으로 본고를 갈음하고자 합니다.

그림7은 OSEK Application 개발 과정을 보여준다. 이 그림에서 System generator는 OIL 을 이용하여 특화된 OSEK OS 와 OSEK COM 을 생성하고 이렇게 생성된 OSEK 모듈들은 응용 프로그램과 함께 링킹되어 최종적으로 하나의 수행파일을 구성한다.

이렇게 만들어진 OSEK OS, COM, NM, OIL을 완벽히 활용하기 위해서는 아직 해결되어야 할 문제들이 많다. 그 중 중요한 것들을 언급하면 다음과 같다. 자동차용 소프트웨어의 전반적인 아키텍처를 설계하는데 사용할 방법론과 이를 지원하는 도구가 고안되어야 한다. 또한 프로그래머들이 소프트웨어컴포넌트들을 손쉽게 개발할 수 있도록 해 주는 그래픽 기반의 개발 환경이 필요하다. 개발 비용을 줄이기 위해 실제 하드웨어가 없는 상황에서도전체 시스템을 시뮬레이션 할 수 있는 테스트 환경이 제공되어야 한다. 자동차 업계 및 학계는 현재 이러한 문제들을 하기 위해 노력하고 있으며 전망은 밝을 것으로 예상한다. [7]

참고문헌

[1] OSEK/VDX Portal: www.osek-vdx.org

[2] OSEK, "OSEK/VDX Operation System Version 2.2.3", 2005.

[3] OSEK, "OSEK/VDX Network Management Version 2.5.3", 2004.

[4] OSEK, "OSEK/VDX Communication Version 3.0.3", 2004.

[5] OSEK, "OSEK/VDX OSEK Implementation Language Version 2.5", 2004.

[6] Joseph Lemieux, "Programming in the OSEK/VDX Environment",

[7] Seongsoo Hong, Jiyong Park, and Wooseok Yoo, "Technology Trends in Automotive OS and Middleware", 2006.

[8] 서영빈, 김상철, 마평수, 최태영, “ROSEK: OSEK기반 자동차용 운영체제”, 2008.