논문 2010-05-13

# 자동차 전장용 실시간 태스크 스케줄링 알고리즘

## (Real-Time Task Scheduling Algorithm for Automotive Electronic System)

권 규 호, 이 정 욱, 김 기 석, 김 재 영, 김 주 만*
(Kyu-Ho Kwon, Jung-Wook Lee, Ki-Seok Kim, Jae-Young Kim, Joo-Man Kim)

Abstract : Due to the increasing amount of electronic control system in a vehicle, the automotive software is increasingly sophisticated and complicated. Therefore it may be faced a time critical problem caused by its complexity. In order to solve such problems, the automotive electronic system can use a real-time scheduling mechanism based on predictability. We first consider the standard specification of the AUTOSAR OS and uC/OS-II such as its scheduling theory with time determinism. In this paper, we propose the scheduling algorithm to be conformable to a conformance class of OSEK/VDX specification. Algorithm analysis shows that our scheduling algorithm outperforms an existing Trampoline OS by intuition.

Keywords : Scheduling, AUTOSAR, OSEK, Real-time, Time-deterministic

## 1. 서 론

자동차의 이동성, 안정성 및 편의성을 제고한 고도의 기술 개발의 결과로 자동차 성능 향상은 물론 탑승자를 위한 편안하고 안락한 첨단 기능을 제공하기에 이르렀다. 이러한 기능들은 다수의 전자제어 장치(ECU)를 통한 지능적 소프트웨어 모듈들의 결합을 통하여 달성 할 수 있다. 그러나 자동차 주행환경의 특수성과 안전성을 위한 시간 기반(Time-triggered) 태스크들에 대한 실시간 특성 고려와 전자적 제어 장치(ECU)의 증가로 인한 분산 태스크 환경의 복잡성은 증대되었다. 따라서 차량의 복잡성 문제에 대한 해결책으로 비동기적 처리가 요구되는 동시 이벤트 처리는 물론, 각 ECU에 다수의 OS-Application이 상호 간섭 없이 수행 할 수 있는 실시간 운영체제가 도입되었으며, 이에 따라 각 자동차 생산 및 부품 업체들이 소프트웨어 개발 환경 및 실시간 운영체제에 대한 규격으로서 AUTOSAR 표준을 제정하게 되었다. AUTOSAR 표준은 현재 4.0 버전이 출시되었으며, 자동차 미들웨어를 구조화 하여 계층에 따라 요구되는 기능을 포함하고 있다. AUTOSAR OS에서 정의하고 있는 자동차 ECU용 OS는 정적 구성, 실시간성, 우선순위 기반 스케줄링, 보호 기능(메모리, 타이밍 등), 하드웨어 제어와 같은 특징을 가진다[10].

AUTOSAR OS의 코아는 OSEK/VDX OS를 기반으로 기존 기능의 확장과 보호 기법 및 시간 기반 시스템을 위한 스케줄 테이블의 도입으로 추가된 기능을 제시하고 있으며, 실시간 태스크를 위한 태스크 개수, 우선순위별 태스크 수, 지원되는 우선순위의 개수 등의 적합성 클래스(Conformance Class)를 두어 이를 지키도록 규정하고 있다[9].

실시간 시스템은 논리적으로 정확하고, 시간을 반드시 지켜야하는 제약 조건을 갖는 시스템으로서 자동차 운행 및 안전과 직결되는 실시간 특성을 구현하기 위하여 반드시 AUTOSAR OS 의 규격을 만족하는 실시간 스케줄링 기법이 요구된다[1][6]. OSEK에서 비롯된 자동차 전장용 운영체제의 성능 향상 연구는 주로 속도 문제와 메모리 효율성 문제를 다루었다[2]. [4]에서는 스케줄링의 속도 성능을 높이는 방법으로 태스크 상태 전이에 따라 문맥 교환을 생략하는 기법으로 효과적인 결과를 얻기도 하였다. 또한 [3]에서는 μC/OS-II를 개선하여 우선순위 개수를 늘이면서 메모리 효율성을 지원하는 다차원 기법을 이용한 실시간 스케줄링 알고리즘을

* 교신저자(Corresponding Author)
논문접수 : 2010. 2. 26., 수정일 : 2010. 03. 22.,
채택확정 : 2010. 04. 26.
권규호,김주만 : 부산대 바이오정보전자공학과
이정욱,김기석,김재영:한국전자통신연구원
※ 이 논문은 부산대학교 자유과제 학술연구비(2년)에 의하여 연구되었음.

연구하였다.

시간 결정성 알고리즘을 채용하고 있는 실시간 운영체제인 µC/OS-II는 각 태스크는 유일한 우선순위만 가질 수 있음으로써 AUTOSAR 의 적합성 클래스 중 BCC2와 ECC2를 따르지 않는다. [8]에서 제안하는 HSE Free OSEK은 µC/OS-II를 OSEK 스케줄링에 그대로 적용한 사례로 들 수 있다.

본 논문에서는 자동차 표준 운영체제의 표준 규약인 AUTOSAR OS 사양을 만족하며, 아울러 시간 결정성(Time Determinism)을 갖는 실시간 스케줄링 알고리즘을 제안한다.

논문의 구성은 2장에서 실시간 스케줄링의 개념 및 본 논문에서 제안하는 알고리즘과 관련있는 µC/OS-II와 전장용 운영체제로 구현된 Trampoline 실시간 운영체제를 소개한다. 3장에서는 AUTOSAR OS의 태스크 개념과 스케줄링 및 표준 명세서를 기술하였고, 4장에서는 시간 결정성 실시간 스케줄링 알고리즘을 제안한다. 5장은 제안된 알고리즘과 AUTOSAR 규격을 구현한 Trampoline 운영체제의 스케줄링 기법과 알고리즘 비교하였으며 6장에서 본 논문의 결론을 맺는다.

## II. 관련연구

### 1. 실시간 스케줄링 개념

실시간 시스템은 시스템 동작의 논리적 정확성을 바탕으로 시간적 정확성을 요구하며, 시간적 제약 조건의 허용의 정도에 따라 경성 실시간(hard real-time)과 연성 실시간(soft real-time) 시스템으로 분류한다. 시간 제약 조건을 만족하기 위하여는 시스템의 모든 활동(태스크, 이벤트 등)이 예측 가능하여야 하기 때문에 시간 결정성이 보장되어야 할 것이다. 실시간 운영체제에서 시간 결정성을 보장하여야 하는 활동은 다음과 같다.

- 태스크 I/O 요청 시간
- 스케줄링시 higher-priority 태스크 검색 시간
- 문맥 교환 시간
- 인터럽트의 반응 속도

이와 같은 활동은 MCU의 성능에 의존적적이지만, 소프트웨어 관점에서 스케줄링 알고리즘을 통한 예측성을 확보하여야 할 것이다. 이러한 예측성의 확보는 알고리즘의 실행 시간이 상수 시간(O(1))이어야 한다. 현재 µC/OS-II 실시간 운영체제가 이를 만족하고 있다.

### 2. µC/OS-II 실시간 운영체제

µC/OS-II는 우선순위 기반의 선점 방식을 지원하는 실시간 운영체제로서 64개의 우선순위를 가지며, 각 태스크에 대하여 준비 리스트 삽입, 삭제 및 문맥 교환을 위한 높은 순위 태스크 선정 시에 시간 결정성을 갖는 알고리즘을 지원한다[7]. 즉, 2차

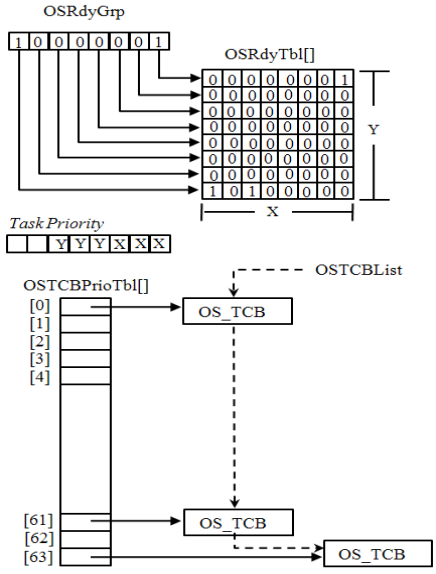


그림 1. µC/OS-II 준비리스트 모형

Fig. 1. Ready list model for µC/OS-II

원 비트 배열인 OSRdyTBL[]이라는 준비 리스트에 준비 상태의 태스크의 비트를 설정하며, 비트 연산을 통하여 상수 수행 시간을 보장한다.

그림1은 태스크 우선순위에 대한 준비 리스트인 OSRdyTbl[]에 적재된 태스크 비트와 실제 태스크 제어 블록인 OS_TCB 가 우선 순위 테이블인 OSTCBPrioTbl[]에 연결된 구조를 보여주고 있다. OSTCBList 는 태스크가 준비 리스트에 삽입되는 순서의 마지막 태스크를 가르키며, 태스크 관리는 비트열의 위치를 결정하는 연산에 의하여 직접 OSTCBPrioTbl[]을 통하여 접근한다.

µC/OS-II의 각 태스크는 유일한 명시자(ID:Identification)를 가지며, 이 ID는 우선순위의 의미로도 사용한다. 즉 µC/OS-II에서 각 태스크는 유일한 우선순위를 가진다. 태스크 우선순위가 유일

하다는 것은 동일한 우선순위를 갖는 타임 트리거(time triggered) 태스크 특성을 반영하지 못하고, 스케줄링 요구 시에 태스크의 서로 다른 우순선위 차이로 인하여 불필요한 문맥 교환을 야기할 수 있다.

### 3. Trampoline 실시간 운영체제

Trampoline 은 자동차용 임베디드 소프트웨어 산업 표준인 AUTOSAR OS의 표준 규격을 구현한 실시간 운영체제이다[5]. 스케줄링을 위하여 각 태스크는 정적 우선 순위로서 규격에 정의된 적합성 등급(Conformance Class)에 때라 여러 개의 태스크가 하나의 우선순위를 공유 할 수 있다. 또한 응용에 따라 완전 비 선점(full non preemptive), 완전 선점(full preemptive) 혹은 혼합 선점(mixed preemptive) 방식으로 스케줄링 된다. Trampoline에서 CPU 사용을 기다리는 태스크는 큐 구조를 갖는 접속 리스트 구조 형식의 준비 리스트에 연결된다.

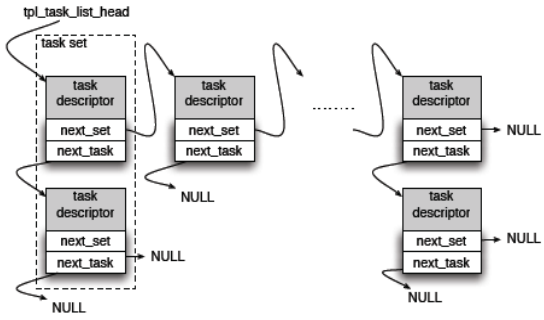


그림 2. Trampoline 준비리스트 모형
Fig. 2. ready list model for trampoline

그림2는 Trampoline OS의 준비리스트 모형을 보여주고 있다. 동일 우선순위를 갖는 태스크 서술자(descriptor)가 접속 구조로 연결되고, 이들 task_set은 높은 우선순위에서 낮은 순으로 next_set으로 연결되어 있다. tpl_task_list_head는 항상 가장 높은 우선순위 태스크를 지시하게 될 것이다. 이러한 구조에서 직관적으로 높은 우선순위 태스크 결정은 항상 복잡도가 O(1) 이지만, 낮은 우선순위 태스크의 활성화나 준비 리스트에서의 삽입과 삭제를 위한 검색은 최악의 경우 O(n)의 시간을 요구한다.

## Ⅲ. 전장용 운영체제 표준 명세서

### 1. 태스크 관리(Task Management)

OSEK/VDX 표준 명세서는 두 종류의 태스크 타입을 제공한다. 베이직 태스크(BT:Basic Task)는 높은 우선순위 태스크로 문맥 교환이나 인터럽트 서비스 루틴 실행이 되기까지 종료 없이 수행되는 순차 코드이다. 따라서 태스크의 시작과 종료 시점에만 동기가 된다. 즉, BT는 다른 이벤트나 높은 우선순위 태스크의 의해 선점 될 수 있으나, 대기상태로 진입하지 않고 수행을 완료하면 종료하는 기본 태스크이다.

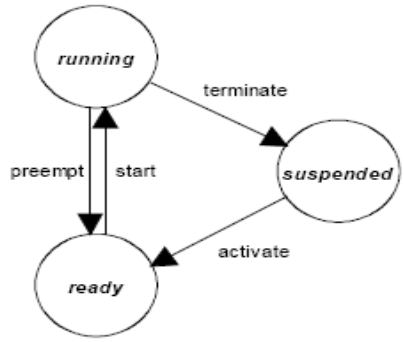


그림 3. BT의 상태 전이도
Fig. 3. state transition diagram for BT

그림3에서 태스크가 준비 상태에서 시작되면 어떤 이벤트나 높은 우선순위 태스크에 의한 선점을 제외하면 종료되기까지 계속 수행되는 모델을 보여주고 있다.

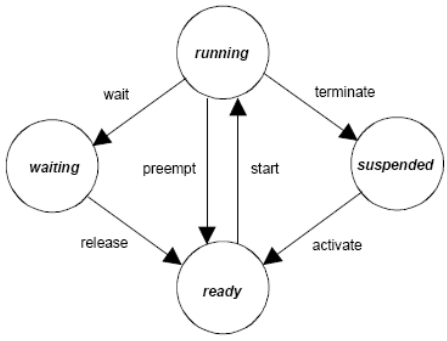


그림 4. ET 상태 전이도
Fig. 4. state transition diagram for ET

또 다른 타입인 확장형 태스크(ET:Extended Task)는 대기 상태를 호출할 수 있음으로 BT와 구

별된다. 대기 상태는 실행중은 현재의 ET를 종료하지 않게 하고 더 낮은 우선순위의 태스크(BT 혹은 ET)에게 처리기를 할당 할 수 있게 한다. 확장형 태스크의 태스크 상태 는 대기(Waiting), 실행(Running), 준비(Ready), 종료(suspended) 상태로 나누어진다.

그림4는 ET의 상태 전이도로서 임의의 서비스 호출을 통한 대기상태로의 전이가 허용됨을 보여주고 있다.

### 2. 스케줄링 정책(Scheduling Policy)

OSEK에서는 인터럽트, 스케줄러, 태스크 레벨등 3가지 처리 레벨을 가진다. 인터럽트 레벨이 가장 높은 처리 순위를 가지며, 태스크 레벨은 낮은 순위에서 사용자에게 부여된 고정 우선순위에 따라 스케줄 된다. 스케줄링 정책은 완전 비 선점(full non preemptive), 완전 선점(full premptive) 혹은 혼합 선점(mixed preemptive)이 될 수 있다.

혼합 선점은 어떤 시점에 선점 혹은 비선점과 같이 적당한 모드를 취할 수 있음을 의미한다. 그룹이라는 의미는 공통 내부 자원들을 공유하기위해 존재한다. 통상적인 선점 기법은 우선순위 레벨에 따라 그룹 내에서는 적용되지 않도록 한다. 즉, 그룹내에서 태스크들 간에는 선점을 허용하지 않는다.

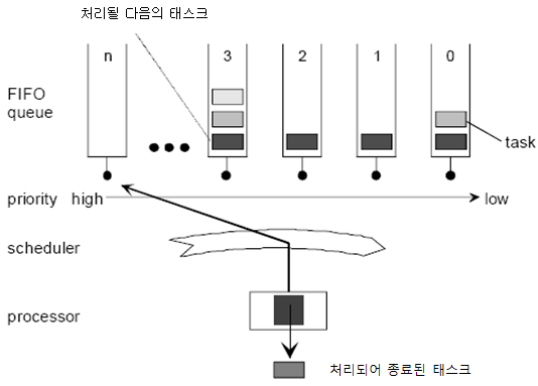


그림 5. OSEK/VDX 스케줄러
Fig. 5. OSEK/VDX scheduler

그림5는 OSEK/VDX 규격에 따라 각 우선순위에 대한 스케줄링 구현 예로서 스케줄러는 각 우선순위별 준비 리스트인 FIFO 큐에서 높은 순위에서 낮은 순위로 태스크를 선정하여 처리하는 과정을 보여주고 있다.

### 3. 적합성 클래스(Conformance Class)

자동차에 다양한 시스템의 성능과 응용 소프트웨어 요구사항을 만족시키기 위해 OS의 다양한 특성을 요구한다. 이런 OS 특성에 따른 분류를 적합성 클래스라고 한다.

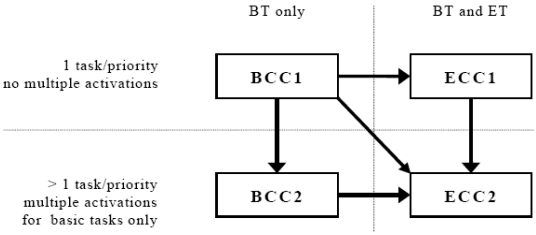


그림 6. 적합성 클래스
Fig.6. Conformance class

적합성 클래스는 태스크 형태, 태스크 활성화의 다중 요청, 우선순위 당 태스크 수 등을 기준으로 결정되는데, 그림6과 같이 BCC1, BCC2, ECC1, ECC2로 나누어진다. BCC1은 BT만 지원하며 각 우선순위마다 하나의 태스크를 가지고 태스크마다 한번 활성화 요청이 이루어지도록 제한되어 있다. BCC2는 BCC1를 포함하고, 추가로 여러 개의 태스크는 동일한 우선순위를 가질 수 있으며, 태스크의 다중 활성화가 허용된다. ECC1은 BCC1에 ET를 지원하며, ECC2는 ECC1을 포함하고, 추가로 여러 개의 태스크는 동일한 우선순위를 가질 수 있으며, 태스크의 다중 활성화가 허용된다. 그림6에서 화살표는 포함관계를 4사분면에 따라 BT와 ET 및 우선순위에 대한 태스크 개수를 규정하고 있다.

## Ⅳ. 시간 결정성 실시간 스케줄링 알고리즘

### 1. 스케줄링 알고리즘 설계 개요

OSEK OS 표준 스케줄러는 준비 상태에서 실행 상태 진행은 태스크 우선순위에 따라 결정된다. 가장 낮은 우선순위의 태스크는 0값을 가지며, 높은 값을 가질수록 우선순위가 높다. 태스크의 우선순위는 정적으로 정의되며 동일한 우선순위를 가질 수 있는 하나 이상의 태스크들은 BCC2와 ECC2에서 지원된다. 스케줄러는 서로 다른 우선순위를 가지는 n개의 큐에서 가장 우선순위가 높은 큐의 준비 상태에 있는 태스크 부터 순차적으로 실행 하게 된다.

본 논문에서는 OSEK/VDX 표준을 만족하면서

수행 시간을 O(1)에 완료하는 시간 결정성 스케줄링 알고리즘을 제안하는데 있다. 시간 결정성 알고리즘은 이미 uC/OS-II에서 구현된 사례가 있으나, 태스크 당 하나의 우선순위만을 지원함으로써 OSEK/VDX 적합성 클래스 BCC2 및 ECC2를 지원하지 못한다. 따라서 본 논문에서는 적합성 클래스를 만족하면서 시간 결정성 알고리즘인 CC2 알고리즘을 제안한다.

## 2. CC2 알고리즘 동작 환경

적합성 클래스를 만족하는 CC2 알고리즘의 동작 환경은 그림7과 같다. 표준에서 최소한으로 요구하는 우선순위 개수인 16단계를 만족하며 적합성 클래스의 BCC2, ECC2에서 요구하는 동일한 우선순위를 갖는 다중 태스크를 지원하기 위해 8개 까지 처리할 수 있는 FIFO 큐를 두었다. 즉, 8개 까지 동일 우선순위를 가질 수 있다. 전체 4개의 그룹을 두었고, 각 그룹당 4개의 FIFO 큐를 두었다. 스케줄러는 그룹의 순위와 그룹내 각 FIFO 큐의 우선순위를 비트 연산에 의한 상수 시간에 다음에 처리할 태스크를 쉽고 빠르게 선정할 수 있다.

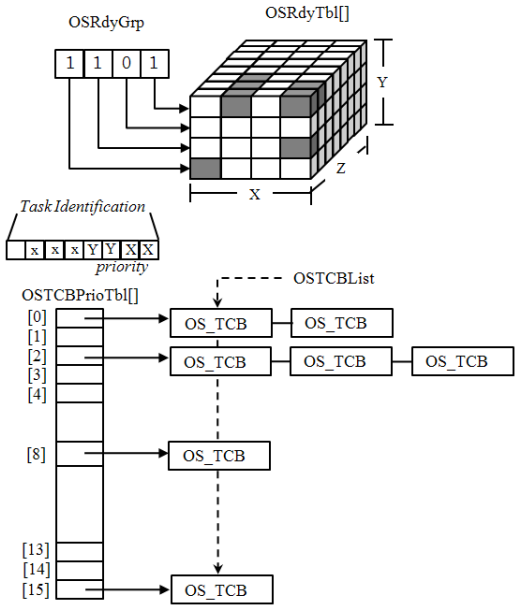


그림 7. CC2 알고리즘 동작 환경
Fig. 7. ready list model for CC2

그림7은 CC2 알고리즘의 동작 환경을 예를 들어 보여주고 있다. 4개의 그룹(OSRdyGrp)에 의해 8개의 동일 태스크를 적재하는 16개의 FIFO 큐로 구성된 준비 리스트(OSRdyTbl[])를 가르키고 있다. 총 태스크 수는 128개까지 지원하며, 7비트의 task_id중에 하위 4비트는 16개의 우선순위를 지시하고, 상위 3비트는 태스크의 고유 ID로서 태스크 생성시에 정의된다. OSRdyGrp과 OSRdyTbl은 OSUnMapTbl을 통하여 가장 높은 우선순위의 태스크를 선정하게 되고, 선정된 태스크의 FIFO 큐를 하향 이동(shift right)하여 동일 우선순위의 다른 태스크가 다음에 선정될 수 있도록 한다. 즉, 3차원 [X][Y][Z] 큐브인 준비 리스트에서 Z=0 인 16개의 태스크에서만 높은 우선순위 태스크를 선정 할 수 있도록 하였다.

실제 태스크 제어 블록(OS_TCB) 는 16개의 우선순위인 OSTCBPrioTbl[] 에 접속하여 선정된 우선순위의 첫 번째 TCB에 접근 할 수 있도록 연결되어진다. 그림7에서 우선순위 0,2,8,15에 각각 태스크가 2,3,1,1 개가 준비 리스트에 있으며, 또한 그 7개의 TCB가 해당 우선순위 큐에 접속되어있음을 볼 수 있다.

## 3. CC2 알고리즘 설계

OSRdyTbl[]은 8개 짜리 동일 태스크를 저장할 수 있는 FIFO 큐가 4개씩, 총 16개로 이루어진 준비 리스트이다. 구현을 위하여 각 우선순위당 8비트 짜리 "char"로 된 FIFO 큐가 4개를 구성하여 하나의 그룹이 32비트 정수가 됨을 알 수 있다. 따라서 32비트 정수를 4개로 분할하여 비트 연산을 용이하도록 다음과 같이 준비리스트를 정의한다.

```
typedef union _OSRdyTblStr {
    long group;
    char fifo[4];
}OSRdyTblStr;

OSRdyTblStr OSRdyTbl[4];
```

그림 8. 준비리스트 자료구조
Fig. 8. ready list data structure

준비 리스트는 4개의 스케줄링 그룹으로 구성되는데, 이 그룹들은 OSRdyGrp 의 4비트에 의해 지시된다. 이들 4비트 이진수에서 LSB가 가장 높은 우선순위를 나타내도록 OSUnMapTbl[]을 두었다.

OSUnMapTbl[]은 16개의 4비트 이진수의 비트중 유효한 LSB 비트의 위치를 나타내고 있다. 이 테이블을 이용하여 가장 높은 우선순위를 갖는 그룹의 위치와 그룹내 가장 높은 우선 태스크를 상수시간에 결정할 수 있다.

```
INT8U const OSUnMapTbl[] = {
    0, 0, 1 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0
}
```

그림 9. 높은 우선순위 결정 테이블
Fig. 9. Higher priority decision table

### 3.1 태스크 삽입 알고리즘

준비 리스트에 새로운 태스크의 삽입은 다음과 같다. 먼저 태스크의 우선순위인 prio의 하위 4비트가 포함하고 있는 그룹 값과 그룹내의 큐의 위치정보를 이용하여, OSRdyGrp에 해당 그룹에 태스크가 있음을 설정하고, 해당 준비 리스트인 OSRdyTbl의 4개 FIFO 큐 중에 해당 큐를 먼저 상향 이동(shift left)하여 태스크 등록됨을 비트 “1”로 설정한다.

```
OSRdyGrp |= OSMapTbl[prio>>2];
if(OSRdyTbl[prio>>2].fifo[prio&0x03])
    OSRdyTbl[prio>>2].fifo[prio&0x03] <<=1;
OSRdyTbl[prio>>2].fifo[prio&0x03]|= 0x01;
```

그림 10. 태스크 삽입 알고리즘
Fig. 10. Task insertion algorithm

### 3.2 태스크 삭제 알고리즘

준비 리스트 해당 FIFO 큐에서 삭제할 태스크 설정 비트를 지운다. FIFO 큐를 하향 이동(shift right)하고, 방금 지운 FIFO 큐가 속한 그룹에 등록된 태스크가 없으면, 그 그룹을 가르키는 OSRdyGrp의 해당 비트를 지운다.

```
OSRdyTbl[prio>>2].fifo[prio&0x03]>>=1;
if(OSRdyTbl[prio>>2].group == 0)
    OSRdyGrp &= ~OSMapTbl[prio>>2];
```

그림 11. 태스크 삭제 알고리즘
Fig. 11. Task deletion algorithm

### 3.3 최상위 우선순위 결정

OSRdyGrp에서 활성화된 비트중 가장 우선순위가 높은 비트를 OSUnMapTbl[]에 의해 구한다. 그 비트가 가르키는 그룹은 8비트 FIFO 큐가 4개인 32비트 길이의 그룹이다. 이들 각 FIFO 큐는 8비트 크기로 항상 LSB로부터 값이 설정된다. 따라서 그룹에 속한 4개의 FIFO 큐에서 LSB 비트만을 추출하여 그 중 가장 높은 우선순위를 갖는 FIFO를 OSUnMapTbl[]을 사용하여 구할 수 있다. 그룹값 y와 그룹내 해당 FIFO 큐 위치 값 x를 조합하여 우선순위 값을 구한다.

```
xbit = 0x00;
y = OSUnMapTbl[OSRdyGrp];
for(i=3; i>=0;i--)
    if(OSRdyTbl[y].fifo[i] & 0x01)
        xbit |= OSMapTbl[i];
x = OSUnMapTbl[xbit];
prio = (y << 2) + x
```

그림 12. 높은 순위 태스크 검색 알고리즘
Fig. 12. Task searching algorithm for highest priority

## Ⅴ. 알고리즘 분석

실시간 스케줄링 알고리즘 설계 시에 대두되는 가장 중요한 이슈는 시간 성능이다. 좋은 시간 성능을 얻기 위해서는 준비 리스트에서 다음에 수행할 태스크를 선정 하거나 준비 리스트에 새로운 태스크를 삽입과 삭제가 상수 시간에 이루어지게 한다면, 스케줄링의 시간 예측성을 보장하게 된다. 스케줄링 성능 개선의 한 방법으로 [5]에서는 태스크 상태 전이에 따라 문맥 교환을 생략하게 함으로써 스케줄링 성능을 개선한 사례도 있다.

AUTOSAR OS를 구현한 Trampoline OS는 준비 리스트를 우선 순위별 FIFO 큐를 두어 양방향 접속 리스트로 구현하였다. 이러한 리스트 구조는 가장 높은 우선순위 태스크를 찾는데 걸리는 시간은 O(1) 이지만, 삽입과 삭제에 걸리는 시간은 최대 O(n)이 소요된다.

본 논문에서 참조한 µC/OS-II 의 스케줄링 알고리즘은 시간 결정성을 보장하지만, AUTOSAR OS

의 적합성 클래스중 BCC2와 ECC2를 지원하지 못함으로써 자동차 전장용 운영체제로의 적용이 불가하다.

본 논문에서 구현한 알고리즘인 CC2는 AUTOSAR OS의 적합성 클래스를 완벽하게 보장함은 물론, 준비 리스트에서의 모든 연산이 상수 시간인 O(1)을 만족 시켜 시간 성능이 우수함을 직관적으로 확인할 수 있다.

## Ⅵ. 결론

본 논문에서는 자동차용 운영체제 표준인 AUTOSAR OS 규격을 따르면서, 스케줄링 시간의 예측성을 보장하는 시간 결정성 실시간 알고리즘 CC2를 제안하였다. 이를 위해 먼저 AUTOSAR 표준 규격에서 명시하는 적합성 클래스를 소개하고, 태스크와 스케줄링 방법론에 관한 표준 규격을 검토하였다. 또한 시간 결정성을 지원하는 µC/OS-II OS와 AUTOSAR OS 규격을 따르는 Trampoline OS와의 직관적 비교를 통하여 제안된 알고리즘 CC2가 우수함을 보였다.

본 연구는 실시간 운영체제의 스케줄링 알고리즘으로서 AUTOSAR OS 표준을 따르는 전장용 운영체제 구현 연구의 시작에 불과하며, 향후 본 연구 결과를 바탕으로 전장용 실시간 운영체제 설계 및 구현연구가 진행 되어야 할 것이다.

## 참고문헌

[1] 권규호, 김기석, 김광수, 김주만, “자동차 전장용 실시간 운영체제의 스케줄링 기법”, 2009 대한임베디드공학회 추계학술발표회, 2009.

[2] 임진택, 금한홍, 박지용, 홍성수, “동적 메모리 사용 감소를 위한 OSEK OS 커널 구현 메커니즘",한국자동차공학회논문지, 제17권, 제3호, pp. 127-141, 2009.

[3] 조문행, 임재석, 이진욱, 김주만, 이철훈, “내장형 실시간 운영체제에서 다차원 기법을 이용한 실시간 태스크 스케줄링 알고리즘", 한국콘텐츠학회논문지, 제10권, 제1호, pp. 94-102, 2010.

[4] Z. Wu, H. Li, Z. Gao, J. Sun and J. Li "An improved method of task context switching in OSEK operating system", Proc. of the 20th Inter. Conf. on Advanced Information Networking and Applications, pp. 217-222, 2006.

[5] J.L. Bechennec, M. Briday, S. Faucou, Y. Trinquet. "Trampoline: an openSource implementation of the OSEK/VDX RTOS specification", IEEE Int. Conf. on ETFA'06,Prague, Czech Republic. pp. 62-69, 2006.

[6] P-E Hladik, A-M Deplanche, S. Faucou and Y. Trinquet "Adequacy between AUTOSAR OS specification and real-time scheduling theory", Inter. Symposium on Industrial Embedded Systems:SIES'07, Lisbon, pp. 225-233, 2007.

[7] Jean J. Labrosse, MicroC/OS-II The Real-Time Kernel 2/E, CMP Books, 2005.

[8] HSE Free OSEK, http://www.hs-esslingen.de/%7Efrruit00/

[9] OSEK Group, "OSEK/VDX operating system specification." http://www.osek-vdx.org

[10] AUTomotive Open Standard ARchitecture, http://www.autosar.org

## 저 자 소 개


**권 규 호(Kyu-Ho Kwon)**

2008년 8월 : 부산대학교 정보통신공학 학사.
현재, 부산대학교 바이오정보전자공학 석사과정.
관심분야 : 임베디드 시스템, 센서네트워크.
Email : kyuhokwon@pusan.ac.kr

**이 정 욱(Jung-Wook Lee)**

1993년 : 경북대 컴퓨터공학과 학사.
1995년 : 경북대 컴퓨터공학과 석사.
현재, ETRI 선임연구원.
관심분야 : 차량전장용 SW 플랫폼, 차량용 네트워크, 자동시스템 생성.
Email : spacedye@etri.re.kr

**김 기 석(Ki-Seok Kim)**

1988년 : 경북대 전자공학과 학사.
1991년 : 경북대 전자공학과 석사.
1991년~현재 : ETRI 책임연구원.
관심분야 : 차량용 SW플랫폼, 모바일 IPTV, 무선 통신, 이동 통신.
Email : kskim@etri.re.kr

**김 재 영(Jae-Young Kim)**

1991년 : 경북대 컴퓨터공학과 학사.
1993년 : 경북대 컴퓨터공학 석사.
1993년~1999년 : LG정보통신(주).
2000년~현재 : ETRI 선임연구원.
관심분야 : 차량용 SW플랫폼, 임베디드 소프트웨어, 유무선 통신.
Email : jaeyoung@etri.re.kr

**김 주 만(Joo-Man Kim)**

1998년 : 충남대학교 컴퓨터공학과 석사.
2003년 : 충남대학교 컴퓨터공학과 박사.
1985년~2000년 : ETRI 책임연구원(운영체제연구팀장)
1995년~1996년 : Novell Inc. Research Center 방문연구원.
2000년~2005년 : 밀양대학교 정보통신공학부 조교수.
2006년~현재 : 부산대학교 바이오정보전자공학과 부교수.
관심분야 : 임베디드소프트웨어, Real-time OS, 분산병렬처리, 고장허용시스템.
Email : joomkim@pnu.edu