OSEK/VDX 운영체제를 위한 태스크 활성화 서비스 설계

권오용*, 임성락**, 유영창***
*호서대학교 메카트로닉스 공학과
**호서대학교 컴퓨터 공학과
***(주)에프에이리눅스
e-mail:lobogelico@naver.com

Design of Task Activation Services for OSEK/VDX

O-Yong Kwon*, Seong-Rak Rim**, Young-Chang Yu***
*Dept of Mechatronics Engineering, Hoseo University
**Dept of Computer Engineering, Hoseo University
***FALINUX co.,Ltd

요 약

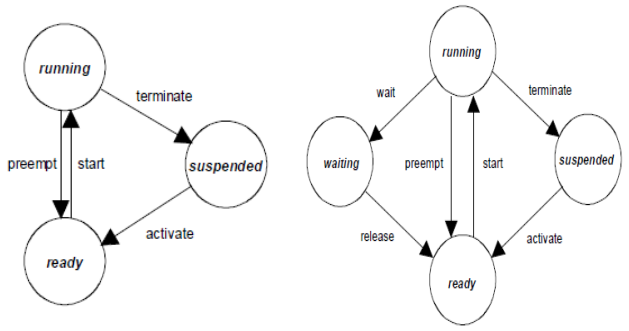
OSEK/VDX 운영체제는 자동차 전자제어장치(ECU)를 위하여 OSEK/VDX에서 제안한 사양을 준수하는 실시간 운영체제이다. 본 연구는 OSEK/VDX 운영체제의 태스크관리 서비스들 중에서 태스크 활성화를 위한 서비스 함수를 설계하고 그 타당성을 제시한다.

1. 서론

OSEK/VDX 운영체제는 자동차 전자제어장치(ECU)를 위하여 OSEK/VDX에서 제안한 사양을 준수하는 실시간 운영체제이다. OSEK/VDX 운영체제의 기본 메커니즘은 태스크를 우선순위에 따라 처리하기 위한 태스크 관리(Task Management), 태스크간의 동기화를 위한 자원 및 이벤트 관리(Resource and Event Management), 경고(Alarm), 카운터(Counter) 그리고 오류처리(Error Handling), 디버깅(Debugging) 기능을 제공한다.[2][3]

2. 태스크 활성화

OSEK/VDX 운영체제는 2가지 종류의 태스크(기본 태스크, 확장 태스크)를 지원하며 태스크의 상태 천이도는 (그림 1)과 같다.



<기본태스크> <확장태스크>

(그림 1) 기본/확장 태스크 상태 천이도

(그림 1)에서 태스크 활성화는 suspended에서 ready 상태로 천이되는 activate 사건에 해당된다. 태스크가 활성화된 후에는 반드시 태스크의 처음 문장부터 실행되어야 한다.[2]

OSEK/VDX 운영체제에서 제공하는 태스크 활성화를 위한 서비스는 ActivateTask(TaskID), ChainTask(TaskID)가 있다. ActivateTask(TaskID)는 TaskID에 해당하는 태스크의 상태를 suspended에서 ready 상태로 활성화 시킨다. ChainTask(TaskID)는 이 서비스를 호출한 태스크의 상태를 running에서 suspended로 전환하고, TaskID에 해당하는 태스크의 상태를 suspended에서 ready 상태로 활성화 시킨다.

2.1 태스크 활성화의 다중요청

태스크 활성화의 다중요청은 이미 활성화된 태스크에 대해 활성화를 중복하여 요청하는 것을 의미한다. OSEK/VDX 운영체제에서는 기본 태스크에서만 활성화의 다중요청을 지원한다. 다중 요청의 최대값은 운영체제 생성시 기본 태스크의 속성으로 부여되며, 최대값에 도달할 때까지 다중요청이 가능하다.

3. 서비스 설계

태스크 활성화를 위한 AcitvateTask(TaskID, ChainTask(TaskID) 서비스의 알고리즘은 (그림 2), (그림 3)과 같다. (그림 2)에서 ActivateTask(TaskID)는 우선적으로, TaskID의 유효 여부를 검사한 후 유효하지 않을 경우, E_OS_ID를 리턴 한다. 다음으로 TaskID를 이용하여

태스크 정보를 얻어온다. 얻어온 태스크 정보에서 활성화 횟수와 정해진 최대 활성화 갯수를 비교하여 최대 활성화 갯수에 도달할 경우 E_OS_LIMIT를 리턴 한다. 만약 최대 갯수에 도달하지 않았을 경우 태스크를 준비큐에 넣고 준비큐내에서 가장 높은 우선순위 태스크를 가져와 다음 실행 태스크로 정한다. 마지막으로 서비스를 호출한 태스크가 비선점일경우 E_OK를 리턴 하여 태스크 스위칭을 진행하지 않는다. 하지만, 선점형 태스크일 경우에 태스크 스위칭을 진행한 후 E_OK를 리턴 한다.

```
ActivateTask(TaskID)
{
  /* TaskID 유효여부 검사 */
  if( TaskID < MAX_TASK)  return E_OS_ID;
  /*  TaskID의 태스크 구조체 얻기 */
      task_info = get_task_id(TaskID);
  /*  활성화 다중 요청 검사 */
  if(task_info.activation > MAX)  return E_OS_LIMIT;
  /*  준비큐에 태스크를 삽입 */
  ready_queue = task_info;
  /* 준비큐내 가장 높은 우선순위 태스크 얻기 */
  next_execute_task = hipriority_task();
  /* 현재 태스크가 선점형 태스크이면 태스크스위칭 한다. */
  if(current_task == PREEMPT) task_switching();
  return E_OK;
}
```

(그림 2) ActivateTask 알고리즘

```
ChainTask(TaskID)
{
  /* TaskID 유효여부 검사 */
  if( TaskID < MAX_TASK)  return E_OS_ID;
  /*  TaskID의 태스크 구조체 얻기 */
      task_info = get_task_id(TaskID);
  /*  활성화 다중 요청 검사 */
  if(task_info.activation > MAX)  return E_OS_LIMIT;
  /*  준비큐에 태스크를 삽입 */
  ready_queue = task_info;
  /*  서비스를 호출한 태스크 종료 */
  terminate_task();
  /* 준비큐내 가장 높은 우선순위 태스크 얻기 */
  next_execute_task = hipriority_task();
  /* 현재 태스크가 선점형 태스크이면 태스크스위칭 한다. */
  if(current_task == PREEMPT) task_switching();
  return E_OK;
}
```

(그림 3) ChainTask 알고리즘

(그림 3)은 ChainTask의 알고리즘을 나타낸 것으로 동작은 ActivateTask와 매우 유사하지만, 서비스를 호출한 태스크를 종료시키는 차이점이 있다.

4. 구현 및 실험

제시한 서비스 알고리즘의 타당성을 확인하기 위해 (주)에프에이리눅스에서 제공하는 EZ-AT7 임베디드 보드[4]에서 <표 1>과 같은 태스크를 사용하여 실험하였다.

<표 1> 실험 태스크

태스크	기능	선점/비선점	우선순위	자동실행
T1	LED-1 ON/OFF	선점	1(Low)	Yes
T2	LED-2 ON/OFF	선점	2	No
T3	화면출력	선점	3(High)	No

T1 만을 활성화하여 동작을 하게 되면 LED-1이 On/Off 된다. 이후 T1에서 ActivateTask(T2)를 호출하여 T2가 활성화 되면 T2가 T1보다 우선순위가 높기 때문에 선점이 일어나 LED-2가 On/Off 됨을 확인 할 수 있다. 다시 T2가 ChainTask(T3)를 호출하면 T2가 종료되고 T3가 화면에 문자를 출력한다. 이후 T3에서 ChainTask(T2)를 호출하여 T2를 활성화 시키면 T3가 종료되고 T2가 LED-2를 On/Off 함을 확인할 수 있다. 이런 실험을 통해 설계한 2가지 서비스가 올바르게 동작하는 것을 확인할 수 있다.

5. 결론

본 연구에서 OSEK/VDX 운영체제를 위한 태스크 활성화 서비스 ActivateTask, ChainTask의 서비스를 설계하고 그 타당성을 제시하였다. 또한 2개의 서비스 함수의 차이점을 제시하고 그에 대한 알고리즘도 같이 제시하였다. 향후과제는 2가지 서비스 이외에 나머지 태스크 관리 함수를 설계하고 그 타당성을 검증하는 것이다.

참고문헌

[1] OSEK/VDK Operating System Specification 2.2.3, 2005. 2. (http://www.osek-vdx.org)

[2] 유우석, 박지용, 홍성수 “분산형 실시간 차량제어 시스템을위한 RTOS, 미들웨어 및 결함 허용성 요소기술연구” 2006.

[3] 서영빈, 김상철, 마평수, 최태영 “ROSEK: OSEK 기반 자동차용 운영체제” 정보처리학회지 제15 권 5호 2008. 9

[4] http://www.falinux.com