실시간 운영체제에서 타임 슬롯을 이용한 소프트 타이머의 설계 및 구현

이재규 °,백대현, 김봉재‡, 정지영†, 이철훈
충남대학교 컴퓨터공학과 °, 삼성탈레스㈜ R&D 팀‡†
{jklee°, dhbaek, chlee}@ce.cnu.ac.kr, {bongjae.kim‡, jyoung.jeong† }@samsung.com

The Design and Implementation of Soft Timer Using Time Slot in Real-Time Operating Systems

Jae-Gyu Lee°, Dae-Hyun Baek, Bon-Jae Kim‡, Ji Young Jeong†, Cheol-Hoon Lee
Dept. of Computer Engineering, Chungnam National Univ. °
Dept. of R&D Team, SAMSUNG THALES CO.,LTD. ‡†

요 약

실시간 운영 체제(Real-Time Operating Systems)는 시스템 동작이 논리적 정확성뿐만 아니라 시간적 정확성에도 좌우되는 운영 체제이다. 시간적 정확성은 결정성(Determinism)이라고도 하며 이를 지키기 위해서는 실시간 운영체제의 제약조건의 하나인 시스템 예측성을 만족해야만 한다. 예측성이란 시스템의 서비스가 정해진 시간 안에 완료되는지를 판별할 수 있다는 것으로 정확하게 동작하는 타이머가 꼭 필요하다. 본 논문에서는 타임 슬롯을 이용해서 타이머의 활성화, 비활성화 루틴을 간단하게 하고 타이머가 만료(Expire)되었음을 간단하게 판별해 낼 수 있는 소프트 타이머를 설계 및 구현 하였다.

1. 서 론

실시간 시스템의 태스크들은 경우에 따라서 일정한 시간 뒤에 스케줄링 되어 동작해야 할 필요가 있다. 예를 들어, 동일한 우선순위의 태스크들에 대해서 라운드 로빈(Round-Robin) 알고리즘을 적용시킬 때, 각 태스크마다 할당된 시간만큼 CPU 를 점유하고 할당된 시간이 지나면 다른 태스크로 문맥 교환(Context-Switch)을 해야 한다. 또한 소프트웨어 기반의 메모리 리프레쉬(Refresh) 메커니즘에서는 일정 시간 간격으로 메모리를 동적으로 리프레쉬 시켜주지 않으면 메모리 데이터에 손실이 발생하게 된다. 이러한 경우들에 있어서 실시간 시스템에서는 미래에 일어날 사건들을 지정해 주어야 한다. 앞으로 수행될 작업을 지정하는 것은 타이머와 타이머 서비스를 이용해서 이루어진다. 이러한 타이머를 구현하는데 있어서 현재 iRTOS™ 에서는 활성화된 타이머의 만료시간(Expire Time)을 델타프로세싱(Delta Processing)에 의해서 관리하고 있다. 델타프로세싱은 활성화된 각 타이머의 만료시간의 차이 값을 유지함으로 만료된 타이머를 판별해 낸다. 이 방법은 활성화된 타이머의 첫 만료 시간만 확인하면 되는 장점이 있지만, 타이머의 활성화와 비활성화시에 타이머 리스트를 탐색하는 작업이 필요하게 된다. 이런 탐색은 운영체제의 예측성을 저해하는 요인이 될 수 있으므로 본 논문에서는 타임 슬롯(Time Slot)을 이용해서 각 타이머의 만료시간과 활성화와 비활성화 루틴을 효율적으로 관리하는 방법을 제시 하였다.

본 논문에서, 2 장은 관련연구로서 실시간 클럭과 시스템 클럭, 타이머 인터럽스 서비스 루틴에 대해서 설명하고, 3 장에서는 타임 슬롯을 이용한 소프트 타이머의 설계 및 구현을 설명하였으며, 4 장에서 테스트 환경 및 결과를 보이고, 5 장에서 결론 및 향후 연구 과제를 기술한다.

2. 관련 연구

2.1 실시간 클럭과 시스템 클럭

실시간 클럭(Real-Time Clock)은 시스템에 내장되어 있으며 시간, 날짜, 연도 등을 처리한다. 이런 실시간 클럭은 배터리로 전원이 공급되는 DRAM 과 함께 구현된다. 따라서 CPU 나 타이머와 상관없이 작동하기 때문에 시스템 전원 상태와 무관하게 시간을 유지할 수 있다. 시스템 클럭은 PIT(Programmable Interval Timer)라는 타이머 칩으로 생성되고, 주로 이벤트 카운터, 경과 시간 알림, 주기성 이벤트 발생, 시간 관련 문제 처리에 사용된다. 타이머 칩은 타이머 컨트롤 레지스터를 가지고 고정된 빠르기의 클럭을 입력으로 받는다. 타이머 인터럽트 빈도는 타이머에서 초당 발생

* 본 논문은 삼성탈레스㈜ 과제 " 다기능 레이더 통제기용 실시간 커널 위탁용역 개발" 의 수행 결과임

하는 인터럽트의 횟수를 의미하며, 입력 클럭을 타이머 칩 내부에서 변화시켜서 얻을 수 있고, 타이머 컨트롤 레지스터를 통해서 설정된다. 타이머 컨트롤 레지스터에 설정하는 타이머 카운트다운 값은 다음 번 인터럽트가 일어날 시간을 결정하며, 입력 클럭이 한 사이클 지나갈 때마다 값이 1 씩 감소한다. 타이머 인터럽트는 틱(Tick)이라고도 불리며 시스템에서 시간의 단위로 사용된다. 예를 들어 타이머가 100 틱짜리이면 1 틱은 10 밀리초가 된다. 시스템 클럭은 주기적으로 이벤트를 발생시킬 수 있기 때문에 실시간 시스템의 동작에 중요한 역할을 한다. 실시간 시스템의 커널에서는 ISR(Interrupt Service Routine)에서 타이머 인터럽트의 발생을 처리하여 커널과 커널 스케줄러에게 알려주고 스케줄러는 틱이 발생할 때마다 적절한 알고리즘으로 태스크를 스케줄링 한다.

2.2 타이머 인터럽트 서비스루틴

타이머 칩을 초기화 할 때 타이머 인터럽트 발생 시 수행될 인터럽트 서비스 루틴을 시스템에 설치해야 한다. 타이머 인터럽트 서비스 루틴에서는 다음과 같은 일을 해야 한다. 시스템 클럭의 갱신, 일정 시간의 경과를 알리기 위해 등록된 함수의 호출, 인터럽트에 대한 응답, 타이머 컨트롤 레지스터의 설정, 인터럽트 처리종료이다.

3. 타임 슬롯을 이용한 소프트 타이머의 설계 및 구현

3.1 타임 슬롯

현재 iRTOS™ 에서는 활성화된 타이머의 만료시간을 델타 프로세싱을 통해서 오름차순으로 정렬한다. 이 방법은 새로 활성화된 타이머가 있을 때 정렬된 리스트에 삽입할 위치를 찾아야 하며, 리스트에서 타이머를 비활성화 시킬 때 삭제할 타이머의 위치를 찾아야 하며 리스트의 만료 타이머 값들을 새로 갱신해야 하는 단점이 있다. 이런 단점을 해결하기 위해서 본 논문에서는 각 활성화된 타이머를 포인팅 하는 배열 10 개를 만들어 이를 타임 슬롯이라고 하고 타이머 인터럽트가 발생할 때 마다 타임 슬롯을 하나씩 이동하도록 해서 만료 시간이 되었음을 판별할 수 있게 하였다.

3.2 타임 슬롯을 이용한 소프트 타이머의 설계

3.2.1 타이머의 생성과 삭제

[표 1]은 타이머를 생성하고 삭제하는 함수를 보여주고 있다.

[표 1] 타이머 생성과 삭제함수

API 함수	설 명
CreateTimer()	타이머를 생성하는 함수
DeleteTimer()	타이머를 삭제하는 함수

3.2.2 타이머 리스트와 제어블록

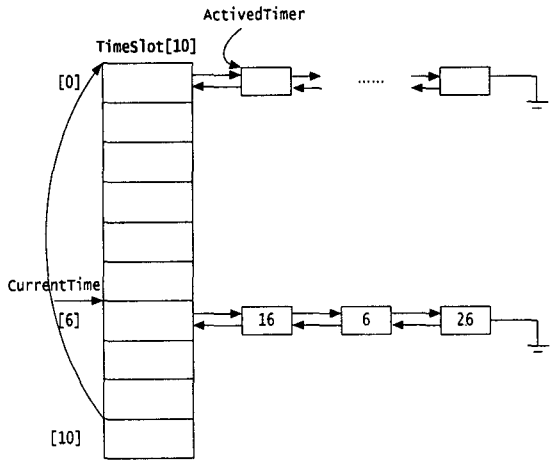
생성된 타이머는 이중 순환 연결리스트로 구성하였으며 새로 생성된 타이머는 리스트의 끝에 삽입되며 삭제 시에는 리스트에서 완전히 제거 된다. 활성화 시에는 이 리스트에서 삭제 하고 타임 슬롯에 연결된다. 타이머의 제어블록은 [표 2]와 같다.

[표 2] 타이머 제어 블록

Field	설 명
tm_Magic	타이머의 식별자(0xF3CD03E9L)
tm_ExpireTime	타이머 수행까지 만료시간
tm_RepeateTime	타이머 수행의 반복시간
tm_TempExpireTime	타임 슬롯의 인덱스 + 타이머의 만료시간
tm_Arg1	타이머 수행 루틴의 첫 번째 인자
tm_Arg2	타이머 수행 루틴의 두 번째 인자
tm_pName	타이머 이름
tm_Function	타이머 수행 루틴
tm_Status	타이머의 상태(활성화 또는 비활성화)
tm_pActivedNext	활성화 시 다음 활성화 타이머의 포인터
tm_pActivedPrev	활성화 시 이전 활성화 타이머의 포인터
tm_pNext	비활성화 시 다음 비활성화 타이머의 포인터
tm_pPrev	비활성화 시 다음 비활성화 타이머의 포인터

3.2.3 활성화된 타이머

[그림 1]에서와 같이 각 타임 슬롯은 활성화된 타이머를 포인팅 하고 있으며 각 타이머의 만료 시간에 따라서 타이머가 연결될 타임 슬롯을 계산하게 된다. 활성화된 타이머가 연결될 타임 슬롯의 인덱스는 (현재 타임 슬롯의 인덱스+활성화된 타이머의 만료시간)%10 으로 결정이 된다. 활성화된 타이머가 연결될 타임 슬롯이 결정 되면 타이머를 타임 슬롯에 연결하고 현재 타임 슬롯의 인덱스+활성화된 타이머의 만료시간을 타이머 제어 블록의 TempExpireTime 에 저장 한다.



[그림 1] 타임 슬롯과 활성화된 타이머

3.2.4 만료된 타이머의 판별

타이머 인터럽트가 발생하게 되면 타임 슬롯의 인덱스를 하나씩 이동 한다. 마지막 9 번 타임 슬롯에서는 다시 0 번 타임 슬롯으로 이동 한다. 즉 타임 슬롯을 한 바퀴 돌면 10 틱이 지난 것이다. 타이머 인터럽트

가 발생해서 다음 타임 슬롯으로 이동하면 슬롯에 연결된 타이머들의 TempExpireTime 이 10 보다 큰지 확인 한다. 10 보다 크면 10 을 뺀 후 ISR 을 마치고 그렇지 않으면 TempExpireTime 과 현재 타임 슬롯의 인덱스 값과 같은지 비교해서 같다면 타이머가 만료 된 것이다. 만료된 타이머의 수행 함수를 수행하고 난 후에 타이머 컨트롤 블록에 반복 시간이 있으면 현재 타임 슬롯에서 제거하고 새로운 TempExpireTime 을 계산해서 새로운 타임 슬롯에 연결 한다.

3.2.5 타이머의 제어

[표 3]은 타이머를 제어하는 함수를 보여주고 있다.

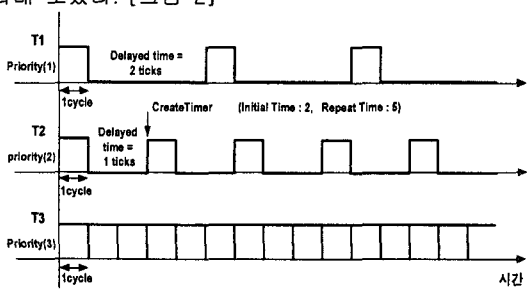
[표 3] 타이머 제어 함수

API 함수	설 명
GetTimerRepeatTime()	타이머의 반복주기를 얻는 함수
SetTimerRepeatTime()	타이머의 반복주기를 설정하는 함수
GetTimerStatus()	타이머의 상태를 얻는 함수
ControlTimer()	타이머의 상태를 변경하는 함수

ControlTimer()함수를 사용해서 특정 타이머를 활성화 또는 비활성화 상태로 만들 수 있다.

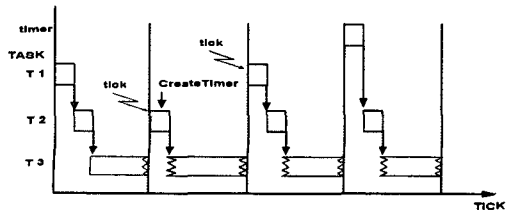
4. iRTOS™ 에서의 테스트 및 결과

본 논문에서 구현한 소프트 타이머를 ARM-920T CPU 가 탑재된 S3C2400 보드에서 동작하는 iRTOS™ 에 적용시켜서 테스트 하였다. 우선순위가 다른 태스크 3 개 T1, T2, T3 를 생성한 후 T2 에서 타이머를 생성하고 활성화해서 다른 태스크들과 동작 타이밍을 나타내 보았다. [그림 2]

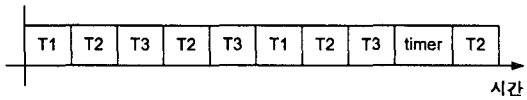


[그림 2] 태스크의 동작 타이밍

위와 같은 태스크들을 생성했을 때 각 태스크들의 동작상태[그림 3]와 실행상태 [그림 4]를 예측해 보면 다음과 같다.

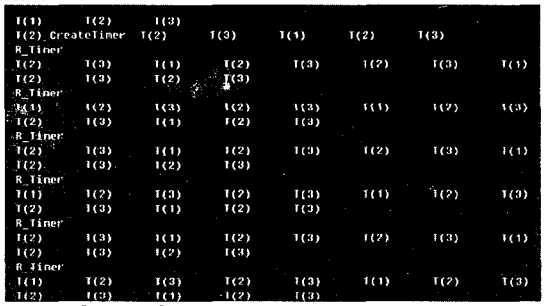


[그림 3] 태스크의 동작상태 예측



[그림 4] 태스크의 실행상태 예측

실제로 테스트 프로그램을 iRTOS™ 커널과 컴파일해서 동작 시켜본 결과 [그림 5]와 같은 결과를 얻을 수 있었으며 앞에서 예측한 결과대로 수행 됨을 알 수 있었다.



[그림 5] 테스트 프로그램 수행결과

5. 결론 및 향후 과제

본 논문에서는 실시간 운영체제에서 시간 결정성을 보장하면서 예측 가능한 소프트 타이머를 설계 및 구현하였다. 타임 슬롯을 이용해서 활성화된 타이머를 관리하고 만료 시간을 판별함으로 타이머의 활성화와 비활성화를 단순화 하였다. 향후 동일한 타임 슬롯에 활성화된 타이머간의 리스트 관리를 좀더 효율적으로 하는 방안에 대한 연구가 더 이루어져야 할 것이다.

6. 참고문헌

[1] http://www.inestech.com
[2] Doudlas Comer, "Operating System Design VOL. I: The Xinu Approach",Prentice Hall, 1988
[3] Alan Burns, Andy Wellings, "Real-Time Systems and Programming Languages", Addison-Wesley, 1996
[4] Jean J. Labrosse " Embedded Systems Building Block, Second Edition Complete and Ready-to-Use Modules in C", CMPBooks, 1999
[5] Qing Li, Caroline Yao, "Real-Time Concepts For Embedded Systems", CMPBooks, 2004