리눅스상의 실시간성 지원을 위한 RTiK-Linux의 설계 및 구현

The Design and Implementation of RTiK-Linux to Support Real-Time on Linux

송 창 인, 김 종 진*, 이 철 훈
충남대학교 컴퓨터공학과, (주)LIG넥스원*

Song chang-in, Kim jong-jin*, Lee cheol-hoon
Dept. of Computer Engineering, Chungnam National Univ., LIG Nex1 Corp.*

요약

최근 빠른 응답성과 정확성을 요구하는 임베디드 시스템의 사용이 증가됨에 따라 임베디드 시스템의 시간 정확성을 만족 시키는 경성 실시간성의 중요성이 커지고 있다. 이러한 임베디드 시스템의 운영체제로는 응용프로그램 개발의 편의성을 위해 범용 운영체제인 Linux를 많이 사용하며, Linux에 실시간성 제공을 위해 RT-Linux(Real Time - Linux)를 사용하고 있다. RT-Linux의 경우 경성 실시간성을 제공하지만 어셈블러를 사용해야 하므로 개발자가 다루기 힘들다는 단점이 존재한다. 이에 따라 Linux에 경성 실시간성을 제공하고 개발자에게 개발의 편의성을 제공하는 방법에 대한 연구가 필요하다. 본 논문에서는 범용운영체제인 Linux에 경성 실시간성을 제공하기 위해 x86기반의 Window에 실시간성을 제공하는 RTiK(Real-Time implanted Kernel)을 Linux에 모듈 형태로 적재하여 실시간성을 제공할 수 있는 방법을 설계 및 구현하였다.

Ⅰ. 서론

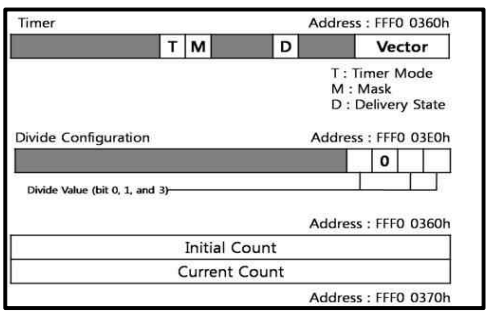
최근 인공위성과 같은 산업용장비에 많이 사용되고 있는 임베디드 시스템의 중요성이 부각됨에 따라, 임베디드 시스템에 빠른 응답성과 정확성을 제공하기 위한 경성 실시간성의 중요성이 커지고 있다. 이에 따라 임베디드 시스템의 운영체제에서 경성 실시간성을 제공하기 위한 연구 개발이 활발히 진행되고 있는 추세이다. 임베디드 시스템의 운영체제는 개발자에게 개발의 편의성 제공을 위해 범용운영체제인 Linux를 많이 사용한다. Linux의 경우 실시간성을 지원을 위해 RT-Linux(Real Time - Linux)를 제공하지만 어셈블러를 사용해야 하므로 개발자가 다루기 힘들다는 단점이 존재 한다. 또한 RT-Linux에서 지원하는 Linux Version이 한정적이라는 단점이 존재하고 있다.

본 논문에서는 이러한 문제점들을 해결하기위해 어셈블러의 사용을 최소화하여 개발자의 편의성을 제공하고, Linux의 Local APIC를 이용하여 경성 실시간성을 지원할 수 있도록 설계 및 구현하였다.

Ⅱ. 관련연구

1. Windows의 RTiK

Windows상에서 실시간성을 제공하는 RTiK은 x86하드웨어에서 제공하는 Local APIC를 이용하여 구현되었다. 그 중 Timer 레지스터의 설정으로 주기적인 인터럽트를 발생시킬 수 있다. [그림 1]는 LVT의 Timer 관련 레지스터에 관한 그림이다. LVT Timer 레지스터는 타이머의 상태정보를 유지하는 레지스터로써 타이머 인터럽트의 모드, 마스크 상태, 인터럽트 전달상태 및 인터럽트 벡터 번호를 가진다. Initial Count 레지스터에 설정한 카운트가 감소되어 0에 도달하면 타이머 인터럽트가 발생하게 된다. Local APIC의 타이머 인터럽트가 발생하면 Timer 레지스터의 Vector에 해당하는 IDT(Interrupt Descriptor Table)의 벡터넘버로 분기하게 된다. IDT에 저장되어 있는 인터럽트의 핸들러 주소를 참조하여 주기적으로 발생되는 타이머 인터럽트의 핸들러를 수행한다. x86기반의 Window에 실시간성을 제공하기위한 디바이스 드라이버 형태인 RTiK을 Linux의 모듈 형태로 적재하여 Linux에 실시간성을 제공할 수 있는 방법을 설계 및 구현하였다[1][2].



▶▶ 그림 1. Timer 관련 레지스터

Ⅲ. RTiK-Linux의 설계 및 구현

Linux의 경우 System Time에 영향을 미치는 Local APIC Timer를 이미 사용하고 있기 때문에 기존의 Local APIC Timer의 핸들러에 영향이 가지 않도록 설계 되어야 한다. 이는 주기 설정에 따른 Local APIC Timer 핸들러가 호출 되어야 한다는 것을 의미한다.

1. IDT(Interrupt Descriptor Table) 등록

Linux의 경우 IRQ0에 대한 event가 존재하기 때문에 기존의 event를 포함하는 IDT vector을 작성해서 동작을 해야 한다. kernel의 apic_32.c파일에서 할당되지 않은 IDT vector와 기존의 IRQ0에 대한 event를 포함한 핸들러를 작성하여 API 통해 인터럽트를 등록 시킨다. 즉, 아래의 [그림 2]에서 보듯이 set_intr_gate()는 인자값으로 벡터 넘버와 처리할 핸들러의 주소를 가지고 IDT에 등록이 된다.

```
Set_intr_gate(0x70,ns_exchange_timer_interrupt);
```

▶▶ 그림 2. IDT의 지정번지에 인터럽트 등록

만약 Local APIC Timer Interrupt가 발생하게 되면 kernel은 event를 가지고 있는 실제 핸들러를 호출하기 위해서 Entry.S 파일을 통해 실질적인 핸들러를 호출 하게 된다. 즉, [그림 3]에서 보듯이 Local APIC Timer Interrupt가 발생 시 set_intr_gate() 를 통해 등록시킨 핸들러가 Entry.S 파일을 통해 ck_ns_exchange_timer_interrupt를 호출하는 것을 확인할 수 있다[3].

```
ENTRY(ns_exchange_timer_interrupt)
        RING0_INT_FRAME;
        pushl $~(nr);
        CFI_ADJUST_CFA_OFFSET 4;
        SAVE_ALL;
        TRACE_IRQS_OFF
        movl %esp,%eax;
        call ck_ns_exchange_timer_interrupt;
        jmp ret_from_intr;
        CFI_ENDPROC;
ENDPROC(ns_exchange_timer_interrupt)
```

▶▶ 그림 3. 핸들러 호출

2. 모듈를 통한 RTiK-Linux 제어

IDT의 256개의 번지 중 비어 있는 번지인 0x70번지에 set_intr_gate() 함수를 사용하여 핸들러를 등록 시켰다. 이에 따라 모듈에서 레지스터 값 설정을 통해 Local APIC Timer Interrupt가 발생하게 되면 벡터 넘버 0x70번째의 핸들러인 ck_ns_exchange_timer_interrupt()를 호출하게 된다. 모듈에서 Timer 레지스터의 벡터 변경, Timer Initial Count 레지스터의 값 및 Timer Current Count 레지스터의 값의 설정을 통해서 설정된 주기로 동작 할 수 있도록 한다[4][5].

Ⅳ. 실험 환경 및 결과

Linux상에 이식된 RTiK을 통해 Timer의 주기적인 동작을 검증하기 위해 [표 1]와 같이 실험환경을 구성하였다.

표 1. 실험환경

실험 환경	
CPU	Intel® Pentium® 4 CPU 2.66GHz
OS	Ubuntu kernel ver 2.6.9
GCC Ver.	Gcc-3.3

RTiK-Linux의 동작 및 주기를 확인 하기위해서 각각의 interrupt가 발생 했을 때의 시간측정을 했다. 이때 정확한 주기 측정을 위해 x86 명령어 집합에서 제공하는 RDTSC(Read Time Stamp Count) 명령어를 사용하여 RTiK의 타이머 인터럽트의 주기를 확인하였다. [표-2]는 0.1ms를 테스트를 하고, 그 테스트 결과를 실험한 컴퓨터의 cpu 클럭 값으로 나눈 표이다. [표-2]에서 보듯이 각 주기를 설정대로 동작하는 것을 확인할 수 있다.

표 2. 0.1ms, 1ms, 10ms 주기 실험결과

RTiK-Linux Test 결과			
	0.1ms	1 ms	10 ms
최대	0.10525 ms	1.00017 ms	10.00001 ms
최소	0.09465 ms	0.99899 ms	9.99984 ms

[표-2]에서 볼 수 있듯이 RTiK-Linux는 설정된 주기를 벗어나지 않고 동작함으로써 Linux에 실시간성을 제공할 수 있음을 확인할 수 있었다.

Ⅴ. 결론

최근 많은 분야에서 실시간성을 이용한 연구가 활발히 진행되면서, 범용운영체제에서의 실시간성 지원이 필수 불가결적인 요소가 되었다. 하지만 Linux의 경우 경성 실시간성이 지원은 되지만, 개발의 편의성을 제공하지 않는 문제점을 가지고 있다.

본 논문에서는 Linux상에 디바이스 드라이버 형태로 구현된 RTiK-Linux를 제공함으로써 x86기반의 하드웨어의 자원 접근 및 제어가 가능하다. 또한 Local APIC를 제어를 통하여 Linux상에서 실시간성을 제공한다. RTiK-Linux의 경우 최소 0.1ms의 주기를 지원하며, 설정된 주기를 벗어나지 않고 동작하는 것을 확인할 수 있었다.

향후 연구과제로는 Local APIC Interrupt handler를 등록하는 것은 모든 Linux버전에 영향을 받지 않지만 어셈블러로 된 실제 Timer 핸들러 코드는 각 Linux버전에 맞게 고쳐주어야 하는데 이것을 수정 없이 패치파일로 제공하여 개발자가 보다 편리하게 사용할 수 있도록 하는 것이다. 그리고 보다 정확한 성능 검증들을 위해 여러 장비에서의 성능에 대한 테스트가 필요로 하다.

▌참 고 문 헌▐

[1] Intel, "Intel 64 and IA-32 Architectures Software Developer's Manual Volume 1 : Basic Architecture", September, 2009.
[2] Intel, "Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3 : System Programming Guide", September, 2009.
[3] Daniel P. Bovet, Marco Cesati "Understanding the Linux Kernel, 3rd edition", 한빛미디어, September,2006
[4] Jonathan Corbett, "Linux Device driver", 한빛미디어, November, 2005
[5] 이귀영, "Linux kernel programming", November, 2001