한국컴퓨터정보학회 동계학술대회 논문집 제20권 제1호 (2012. 1)

멀티프로세서 윈도우즈 XP 상에서 실시간성 지원

송창인°, 이승훈*, 이철훈*
°*충남대학교 컴퓨터공학과
e-mail:{jjami1234, sh2011, clee}@cnu.ac.kr

Real-Time Support on Windows XP of Multi-Processor

Chang-in Song°, Seung-hoon Lee*, Cheol-Hoon Lee*
°*Dept. of Computer Science, Chung-Nam University

● 요 약 ●

실시간성(Real-Time)이란 정확한 주기에 정확한 결과 값의 반환, 혹은 일정한 주기마다 정해진 일을 수행하는 것으로써, 내장형 소프트웨어 분야에서 갖추어야할 가장 중요한 요건이다. 최근 이러한 내장형 시스템의 사양이 싱글프로세서에서 멀티프로세서로 변화됨에 따라 멀티프로세서 상에서의 실시간성을 제공하기 위한 연구가 중요시 되고 있다. 실시간성은 운영체제에 종속적으로 운영체제에 따라 제공여부가 결정되는데, 범용 운영체제인 리눅스의 경우 운영체제에 패치(Patch)를 통해 실시간성을 제공하고 있다. 그러나 다른 범용 운영체제인 윈도우즈의 경우 고가의 구입비용을 가지는 서드파티를 구매하여 실시간성을 제공해야한다. 이에 따라 실시간성을 제공하는 윈도우즈 기반의 제품을 개발 시 개발 비용의 증가를 초래하는 문제점이 발생한다. 따라서 이러한 문제점을 해결하기 위해 범용 운영체제인 멀티프로세서 기반의 윈도우즈 XP 환경에서 실시간성을 제공하기 위한 연구가 필요하다.

키워드: 실시간성(Real-Time), 윈도우즈(Windows), 내장형 시스템(Embedded System)

I. 서론

최근 내장형 시스템의 하드웨어 사양이 싱글프로세서에서 멀티프로세서로 변화됨에 따라 멀티프로세서 기반에서 호환될 수 있는 시스템 프로그램의 중요해지고 있는 실정이며, 실시간성의 제공 여부가 부각되고 있다. 실시간성이란 정확한 주기에 정확한 결과 값의 반환, 혹은 일정한 주기마다 정해진 일을 수행하는 것으로써, 운영체제에 종속으로 운영체제의 실시간성 제공여부에 따라 결정된다. 범용 운영체제인 리눅스의 경우 RT-Linux나 RTAI와 같이 커널 패치 파일을 제공하여 실시간성을 제공한다. 그러나 RT-Linux와 RTAI는 리눅스 커널 버전에 제한적으로 제공되는 문제점이 존재한다. 다른 범용 운영체제인 윈도우즈의 경우, 실시간성을 제공하지 않기 때문에 윈도우즈에 실시간성의 제공하기 위해서는 RTX나 INTime과 같은 서드파티를 사용해야 한다. 그러나 서드파티들은 고가의 구입비 및 경상 사용료를 요구하기 때문에 제품 개발 시 개발 비용이 증가하는 문제점이 있다. 따라서 x86 기반의 멀티프로세서 기반의 윈도우즈 환경에서 실시간성을 제공하기 위한 연구가 필요하다.

본 논문에서는 x86기반의 멀티프로세서의 윈도우즈 XP 상에서 실시간성을 제공하기 위한 RTiK-MP(Real Time implanted Kernel - Multi Processor)를 설계 및 구현하였다. 또한 실험 결과를 통해 멀티프로세서 기반의 윈도우즈 XP 상에서 실시간성을 제공할 수 있음을 증명한다.

본 논문은 2장에서 관련 연구로 윈도우즈에 실시간성을 제공하는 서트파티와 멀티프로세서 기반의 Local APIC(Advanced Programmable Interrupt Controller)에 대해 기술하고, 3장에서 멀티프로세서 기반의 윈도우즈에 실시간성을 제공하기 위한 실시간 이식커널의 설계 및 구현에 대한 설명과 실험결과에 대해 기술한다. 마지막으로 4장에서 결론 및 향후 연구과제에 대해 기술한다.

II. 관련 연구

1. 관련연구

1.1 서드파티

IntervalZero의 RTX(Real-Time Extension)는 윈도우즈 XP 및 윈도우즈 7에 실시간 제어가 가능한 실시간 운영체제의 기능을 부가해주는 확장 소프트웨어로서, 대부분의 개발자들이 익숙하고 편리한 윈도우즈 환경에서 RTX를 이용하여 기존의 실시간 운영체제의 장점을 최대한 사용할 수 있도록 제공해 준다. 즉, RTX는 순수 실시간 운영체제가 아니라 윈도우즈의 대중성 및 풍부한 GUI Library의 장점을 최대한 이용하여 실시간성 등의 제약사항들을 보완해 주는 소프트웨어이다. 이러한 RTX는 멀티프로세서 기반의 윈도우즈에 실시간성을 제공하기 위해 Dedicate Mode를 사용하여

하나의 CPU를 RTX가 독점하여 실시간성을 제공해준다. 이때 독점된 CPU에서는 윈도우즈의 일반 응용프로그램들은 동작하지 않으며, 오직 RTX의 실시간성을 제공하는 프로그램만 동작하게 된다. 마찬가지로 RTX의 실시간성을 제공하는 프로그램은 윈도우즈가 동작하는 CPU에서 동작하지 않는다. Dedicate Mode를 통해 실시간성을 제공하는 RTX는 현재 국방, 항공, 계측기, 시뮬레이션, 의료기기 등의 많은 산업분야에서 사용되고 있다[1].

1.2 멀티프로세서의 Local APIC

APIC는 인텔 IA-32 혹은 IA-32e 아키텍처 프로세서가 가지는 프로그램 가능한 인터럽트 컨트롤러서, 인텔 프로세서가 제공하는 Local APIC는 프로세서의 인터럽트 핀의 신호, 혹은 APIC 타이머, 외부 장치로부터 받은 인터럽트를 프로세서에 전달함으로써 프로레서가 인터럽트를 처리하도록 하는 역할을 한다. 최근 컴퓨터의 사양이 멀티프로세서로 변화되고, CPU의 수가 증가함에 따라 Local APIC도 각각의 프로세서마다 존재하게 되었다. 이에 따라 각각의 CPU에 존재하는 Local APIC에서 인터럽트를 발생시킬 수 있게 되었다. Local APIC는 여러 개의 APIC 레지스터의 집합으로 구성되어 있으며, 각각의 APIC 레지스터를 이용하여 인터럽트를 발생 시킬 수 있다. 아래의 그림 1은 Local APIC 타이머 인터럽트에 관련된 레지스터로써, 이를 이용하여 윈도우즈와는 독립적인 인터럽트를 발생 시킬 수 있다. 그림 1의 첫 번째의 Timer 레지스터는 Timer 인터럽트가 한 번 발생되는 One Shot mode인지 주기적으로 발생하는 Periodic Mode인지를 결정하는 모드 비트와 Timer 인터럽트를 마스킹 하거나 언 마스킹하기 위한 마스크비트, 인터럽트 처리 루틴에 대한 IDT(Interrupt Descriptor Table)에서의 인덱스를 나타내는 벡터 필드를 포함하고 있다. Timer의 주기는 Initial Count 레지스터의 값에 의해 결정이 되며, Current Count 레지스터의 값이 0이 되면 Timer 인터럽트가 발생 시키고 Initial Count 레지스터의 값이 다시 Current Count 레지스터에 등록됨으로써 주기적인 동작을 수행하게 된다[2-6].

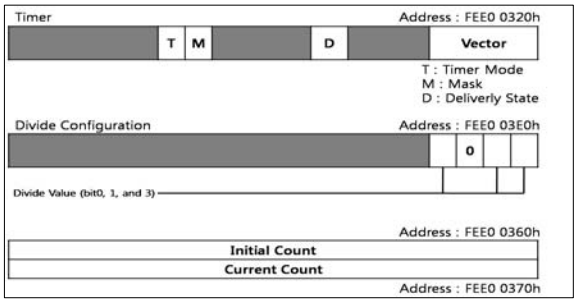

그림 1. Local APIC Timer Interrupt 관련 레지스터
Fig. 1. Local APIC Timer Interrupt registers

III. 본론

1. 본론

1.1 멀티프로세서 윈도우즈 상에서 실시간성 지원을 위한 설계 및 구현

멀티프로세서 윈도우즈 XP 상에서 실시간성을 제공하기 위해서 본 논문에서 구현한 RTiK-MP는 앞서 기술한 Local APIC Timer 인터럽트를 이용하였다. 멀티프로세서중 하나의 프로세서를 선택하여 해당 프로세서의 Local APIC에 접근하여, 주기 설정 및 타이머 인터럽트가 발생 하도록 구현하였다. 그림 2는 프로세서가 두 개일 경우 임의의 프로세서를 선택하여 프로세서의 Local APIC에 접근하는 것을 나타낸 그림이다.

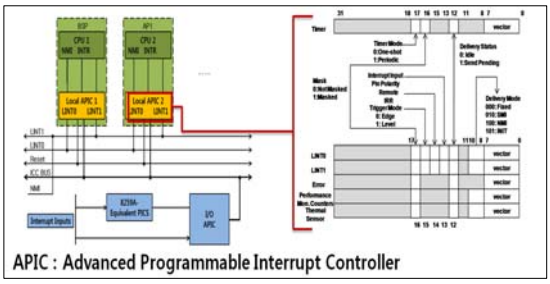

그림 2. 멀티프로세스 상에서 Local APIC
Fig. 2. Multi-Processor on Local APIC

Local APIC를 통하여 Timer 인터럽트가 발생되면 Local APIC의 Timer 레지스터의 벡터 값을 이용하여 IDT로 분기하게 된다. IDT에는 해당 인터럽트의 처리루틴의 주소 값을 가지므로 이것을 통해 발생한 인터럽트를 처리한다. 따라서 Timer 인터럽트를 이용하여 주기적인 작업을 처리하기 위해서는 IDT에 미리 Timer 인터럽트에 해당하는 처리루틴을 등록해야한다. 이를 위해 윈도우즈 XP 상에서 동작하는 RTiK-MP의 경우 윈도우즈에서 제공하는 API를 이용하여 IDT의 벡터 값과 인터럽트가 처리될 프로세서를 지정하는 CPU Affinity 값을 얻어와 인터럽트 처리 루틴과 인터럽트 벡터를 연결하고, 해당 프로세서의 IDT에 인터럽트 처리 루틴과 연결한 벡터를 등록하였다. 그림 3은 윈도우즈의 API를 통해 IDT의 벡터 값과 CPU Affinity의 값을 얻어오는 코드이다.

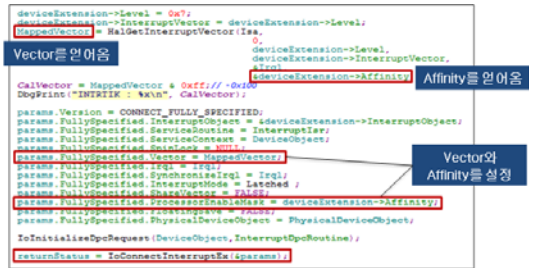

그림 3. 인터럽트 등록
Fig. 3. Interrupt register

그림 3과 같이 윈도우즈에서 제공하는HalGetinterruptVector() API를 사용하여 윈도우즈가 사용하지 않는 IDT의 벡터 넘버와 인터럽트가 처리될 프로세서의 Affinity를 얻어오고, 인터럽트 백터와 인터럽트 처리 루틴, 프로세서의 Affinity를 인자 값으로 하는 IoContectInterruptEx() API를 사용하여 인터럽트가 처리될 프로세서의 IDT에 인터럽트 벡터를 등록하도록 구현하였다. 이를 통해 인터럽트가 발생이 되면 해당 프로세서의 IDT를 참조하여 인터럽트의 처리루틴이 실행하게 된다[7].

1.2 실험 환경 및 실험 결과

RTiK-MP가 윈도우즈 XP 상에서 정상 이식되어 실시간성을 보장하는 것을 검증하기 위해 구성한 실험 환경은 표-1과 같다.

표 1. Host와 Target 컴퓨터의 실험환경
Table 1. System Environment

실험환경		
	HOST	TARGET
CPU	Intel Pentium 4 3.00GHz	Intel Core2Duo 2.33GHz
Operating System	Windows XP SP3	Windows XP SP3
DDK	WDK 7600.163851.1	WDK 7600.163851.1
Development Tool	Ms Visual Studio 2005	WDK Checked Builder

Target 컴퓨터에 RTiK-MP를 이식하고, 정상적인 동작을 확인하기 위해 Host 컴퓨터와 Target 컴퓨터를 시리얼 포트로 연결하여 Host 컴퓨터에서 디버깅 및 모니터링 하였다. 실험 방법은 Target 컴퓨터에서 Local APIC 타이머 인터럽트를 발생시켜 IDT에 등록한 인터럽트의 처리루틴이 실행될 때 마다 x86어셈블러인 RDTSC를 이용하여 실행 주기를 측정하고, 측정된 실행 주기 결과 값을 Host 컴퓨터를 통해 확인하였다. 그림 4-6은 타이머 주기 설정을 0.1ms와 1ms, 10ms로 설정한 뒤 인터럽트 처리 루틴을 10,000번을 수행시킨 뒤 주기 값을 측정한 결과이다.

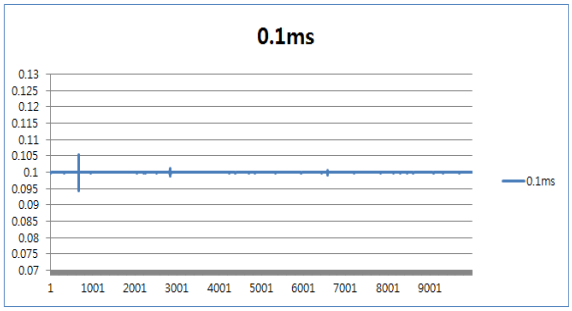

그림 4. 0.1ms 주기 실험결과
Fig. 4. results of 0.1ms-cycle test

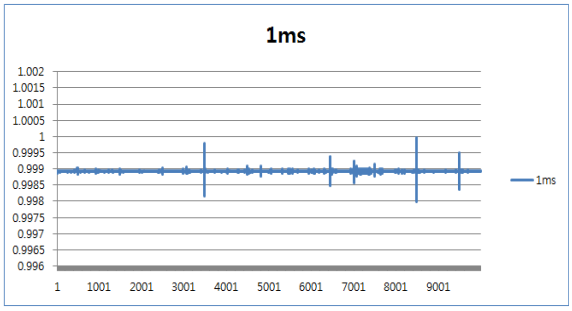

그림 5. 1ms 주기 실험결과
Fig. 5. results of 1ms-cycle test

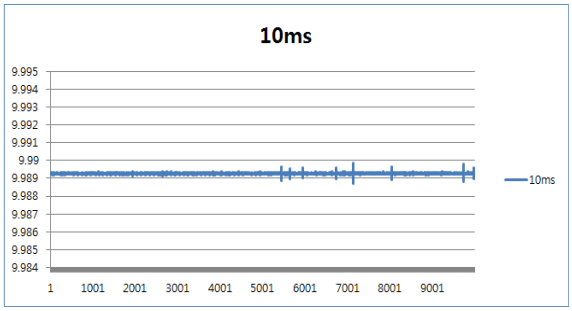

그림 6. 10ms 주기 실험결과
Fig. 6. results of 10ms-cycle test

표 2. 타이머 주기 0.1, 1, 10ms의 실험 결과표
Table 2. result table of Timer-cycle 0.1, 1, 10ms test

	0.1ms	1ms	10ms
최대값	0.1055ms	0.9999ms	9.9898ms
최소값	0.0942ms	0.9979ms	9.9887ms

표-2에서 볼 수 있듯이, 동일한 조건에서 RTiK-MP의 0.1ms, 1ms, 10ms를 측정했을 때 각각 최대 5%, 0.01%, 0.01% 이내의 오차를 가지는 것을 알 수 있다. 위의 결과를 통해 RTiK-MP가 설정한 주기 값을 크게 벗어나지 않고 동작하는 것을 확인할 수 있다. 또한 설정한 주기 값이 커질수록 동작하는 주기가 설정한 주기 값에 근접하는 것을 알 수 있다. 따라서 RTiK-MP가 윈도우즈 XP 상에서 정확한 주기성을 통해 실시간성을 제공하는 것 확인할 수 있다.

IV. 결론

최근 실시간성을 요구하는 내장형 시스템의 사양이 고성능의 높은 사양으로 변화함에 따라 프로세서의 사양 역시 싱글 프로세서에서 멀티프로세서로 변화되었다. 이에 따라 멀티프로세서 기반의 내장형 시스템에서 실시간성을 제공하기 위한 연구가 필요하다. 내장형 시스템의 경우 개발의 편의성을 위해 범용운영체제를 많이 사용하는데 Linux의 경우 실시간성을 제공하는 커널 Patch

파일이 커널 버전에 제한적이고, 커널 Patch를 통한 불편한 설치를 요구하는 단점이 존재한다. 또한 윈도우즈의 경우 윈도우즈가 실시간성을 제공하지 않으므로 실시간성을 제공하기 위해서 고가의 서드파티를 구매해야한다. 그러나 서드파티를 이용하여 제품 개발 시 개발 비용이 증가하여 가격 경쟁에 불리한 단점이 존재하였다. 따라서 본 논문에서는 멀티프로세서 기반의 범용 운영체제인 윈도우즈에 실시간성을 제공하기 위해 윈도우즈의 Local APIC의 Timer 관련 레지스터와 윈도우즈 API를 사용하여 RTiK-MP를 설계 및 구현하고, 실험 측정 결과를 통해 RTiK-MP가 실시간성을 보장하는 것에 대한 검증을 하였다. 이를 통해 고가의 서드파티를 대체하여 가격 경쟁에서 우위를 선점할 수 있다.

향후 연구과제로는 우선순위 기반의 실시간 쓰레드 스케줄러를 구현함으로써 다수의 실시간 쓰레드의 우선순위에 대한 동작을 보장하는 것이다. 또한 RTiK-MP와 사용자영역과의 통신을 통해 사용자 영역에 실시간성을 제공하기 위한 방안 연구가 필요하다. 마지막으로 다양한 API를 제공함으로써 개발자로 하여금 보다 편리하게 RTiK-MP을 사용하도록 하는 연구와 여러 장비에서의 성능 검증이 필요하다.

참고문헌

[1] RTX http://www.intervalzero.com

[2] Microsoft Robotics Studio, Microsoft

[3] Intel, “Intel I/O Controller Hub 6(ICH6) Family Datasheet”, January, 2005.

[4] Intel, “Intel 64 and IA-32 Architectures Software Developer’s Manual Volume 3 : System Programming Guide”, September, 2009.

[5] Intel, “Intel 64 Architectures x2APIC Specification”, June, 2008.

[6] Intel, “Intel 64 and IA-32 Architectures Software Developer’s Manual Volume 2 : Instruction Set Reference”, September, 2009.

[7] Microsoft Windows Sysinternals, http://technet.microsoft.com/en-us/sysinternals/default.aspx