http://dx.doi.org/10.5392/JKCA.2013.13.11.541

태블릿 PC 환경의 실시간 처리 기능 지원

Real-Time Support on the Tablet PC Platform

박지윤*, 조아라*, 김효중**, 최정현**, 허용관**, 조한무**, 이철훈*
충남대학교 컴퓨터공학과*, LIG 넥스원**

Ji-Yoon Park(pjy2013@cnu.ac.kr)*, Ah-Ra Jo(ahrajo@cnu.ac.kr)*,
Hyo-Joung Kim(hyojoung.kim@lignex1.com)**,
Jung-Hyun Choi(junghyun.choi@lignex1.com)**,
Yong-Kwan Heo(yongkwan.heo@lignex1.com)**, Han-Moo Jo(hanmoo.jo@lignex1.com)**,
Cheol-Hoon Lee(clee@cnu.ac.kr)*

요약

태블릿 PC의 경우 개발의 편의성 및 다양한 기능을 제공하기 위해 모바일 운영체제인 윈도우 8을 사용하는데 윈도우 계열의 경우 실시간 처리를 보장하지 못하는 문제점이 있다. 또한 기존의 상용 솔루션과 RTiK 계열의 경우 윈도우와는 독립적인 타이머 인터럽트를 생성하기 위해 사용했던 로컬 APIC 타이머 카운트 값을 얻어 올 수 없기 때문에 실시간 처리 기능을 제공하기 어려운 문제점이 있다. 따라서 본 논문에서는 태블릿 PC의 윈도우 8환경에 실시간 처리 기능을 제공하기 위해 MSR_FSB_FREQ 레지스터를 이용하여 로컬 APIC 초기 카운트 값을 설정하였다. 또한 윈도우의 저전력 기법인 C-State를 제어함으로써 생성한 타이머 인터럽트의 주기성을 보장하여 실시간 처리 기능을 제공하는 RTiK+를 설계 및 구현하였다. 구현한 RTiK+의 성능 검증 및 평가를 위해 CPU 클럭 틱의 수를 반환하는 RDTSC 명령어를 사용하여 생성된 실시간 쓰레드의 주기를 측정하였고, 1ms 주기에서 오차범위 내에서 정상 동작함을 확인하였다.

■ 중심어 : | 태블릿 PC | 윈도우 8 | 실시간 운영체제 |

Abstract

Generally in case of tablet PC's, the Windows 8 is used to support various functions or development convenience, however it cannot support real-time processing. In addition, existing commercial solutions and RTiK has a problem to support real-time processing due to impossibility of getting APIC timer count value which is used to generate timer interrupt separated from that of Windows. Thus, in this paper, we set the initial APIC count value using MSR_FSB_FREQ to support real-time processing on the Windows 8-based tablet PC's. Additionally, we deal with designing and implementing RTiK+ providing real-time processing to guarantee interrupt periods by controlling C-State which is used for low power techniques. To evaluate the performance of the proposed RTiK+, we measured the periods of generated real-time threads using RDTSC instructions which return the number of CPU clock ticks, and verified that RTiK+ operates correctly within the error ranges of 1ms.

■ keyword : | Tablet PC | Windows 8 | Real-time Operating System |

접수일자 : 2013년 10월 07일
수정일자 : 2013년 11월 20일
심사완료일 : 2013년 11월 20일
교신저자 : 이철훈, e-mail : clee@cnu.ac.kr

I. 서 론

최근 임베디드 시스템이 발달함에 따라 일상생활에서 사용되는 태블릿 PC와 스마트 폰에 실시간 처리 기능을 부가한 연구가 활발히 진행되고 있다. 대표적인 예로 군사강국인 미국의 경우 이미 무인항공기에서 촬영한 동영상을 태블릿 PC에 실시간으로 전송하는 감지 기술을 개발하는 등 태블릿 PC를 이용한 다양한 기술 개발 및 성능 평가를 진행하고 있다. 이러한 군사용 태블릿 PC 및 스마트 폰은 무기체계의 실시간 점검 및 데이터 전송을 목적으로 하고 있기 때문에 실시간 처리 기능의 제공을 필수적으로 요구하고 있다. 이러한 실시간 처리 기능의 경우 무인 항공체계 및 유도 무기체계, 의료기기와 같은 다양한 분야에 적용시키기 위한 연구가 활발히 진행 중이다. 하지만 일반적으로 태블릿 PC의 경우 개발의 편의성 및 다양한 기능을 제공하기 위해 모바일 운영체제인 윈도우 8을 사용하는데, 윈도우 운영체제는 실시간 처리 기능을 제공하지 못하는 문제점이 있다. 따라서 태블릿 PC에 실시간 처리 기능을 제공하기 위해서는 상용 솔루션을 사용해야한다. 현재 많이 사용되고 있는 상용 솔루션인 IntervalZero사의 RTX(Real Time Extension)와 tenAsys사의 INtime의 경우 이미 윈도우 8에 실시간 처리 기능을 제공하기 위한 연구가 진행되어, 64bit용 윈도우 운영체제와 윈도우 8환경에 실시간 처리 기능을 제공하지만 고가의 구입 비용 및 경상 사용료로 인한 개발비용 및 유지보수 비용의 증가를 초래한다. 더불어 윈도우 계열에 실시간 처리 기능을 제공하기 위해 기존에 개발된 RTiK(Real-Time implant Kernel) 및 RTiK-MP (Real-Time implant Kernel-Multi Processor)의 경우 윈도우에 독립적인 타이머를 생성하기 위해 사용했던 로컬 APIC(Advanced Programmable Interrupt Controller) 타이머 카운트 값을 윈도우 8에서는 얻어올 수 없기 때문에, 실시간 처리 기능을 제공하기 어려운 문제점이 있다[1][2].

본 논문에서는 이러한 문제점을 해결하기 위해 태블릿 PC의 윈도우 8환경에 실시간 처리 기능을 제공하기 위한 연구를 진행하였다. 이를 위해 CPU의 동작 주파수를 결정짓는 MSR_FSB_FREQ 레지스터를 제어하여 로컬 APIC 타이머 카운트 값 설정과 Intel에서 제공하는 CPU 전력 관리기술인 C-State를 제어함으로써 윈도우와는 독립적인 타이머 인터럽트를 발생시키는 RTiK+를 설계 및 구현하였다. 또한 구현한 RTiK+를 태블릿 PC에 이식하여, 최소 1ms의 주기에서 정확한 응답속도로 동작하는 것을 증명함으로써 실시간 처리 기능을 제공하는 것을 확인하였다.

본 논문은 2장에서 관련연구로써 윈도우에 실시간 처리 기능을 제공하는 RTiK-MP과 저전력 기법에 대해 기술하고, 3장에서 윈도우 8환경의 태블릿 PC에 실시간 처리 기능을 제공하기 위한 RTiK+의 설계 및 구현에 대해 설명한다. 4장에서는 실험환경 및 결과를 기술하고, 마지막 5장에 결론 및 향후 연구과제에 대해 기술한다.

II. 관련연구

현재 윈도우 운영체제에 실시간 처리 기능을 제공하기 위해 개발된 RTiK-MP에 대해 기술하고, Intel에서 윈도우에 저전력 기술을 제공하기 위한 C-State에 대해 기술한다.

1. RTiK-MP

1.1 RTiK-MP의 구성

x86 기반 멀티코어 환경의 윈도우에서 실시간 처리 기능을 제공하는 RTiK-MP의 경우 [그림 1]과 같이 디바이스 드라이버 형태로 이식되어 윈도우의 커널 자원 및 하드웨어 자원을 이용할 수 있다. Real-Time HAL Extension에서 x86 멀티코어 하드웨어를 접근 및 제어하고, 윈도우 커널에 RTiK-MP에서 제공하는 API를 이용하여 실시간 쓰레드를 생성하고 관리한다. 개발된 RTiK-MP의 경우 현재 유도무기체계를 위한 휴대용 점검장비에 이식되어 사용되고 있으며, 다양한 분야에 실시간 처리 기능을 제공하기 위해 통신 미들웨어를 개발하는 등 연구가 활발히 진행되고 있다.

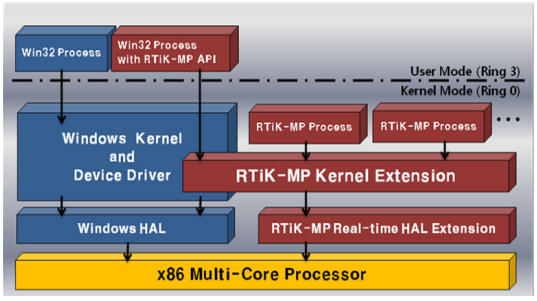

그림 1. 멀티코어 기반의 실시간 윈도우 운영체제 전체 구성도

1.2 RTiK-MP의 동작과정

RTiK-MP는 윈도우에 실시간 처리 기능을 제공하기 위하여 로컬 APIC를 이용해 윈도우와는 독립적인 타이머 인터럽트를 발생시킨다. [그림 2]에서 볼 수 있듯이 RTiK-MP는 BSP(Boot Strap Processor)가 아닌 AP(Application Processor)의 로컬 APIC를 이용하여 윈도우와는 독립적인 타이머 인터럽트를 발생시켜 실시간 쓰레드의 주기적인 동작을 보장한다. 로컬 APIC 타이머 인터럽트가 발생되면 AP의 IDT(Interrupt Descriptor Table)에 인덱스 역할을 하는 벡터를 통해 IDT에 등록되어 있는 인터럽트 핸들러로 분기하게 된다. 이때 수행되는 핸들러는 인터럽트 지연시간을 최소화시키기 위해 지연처리함수(DPC:Deferred Procedure Call) Queue에 등록된다. 인터럽트 핸들러가 완료되고, IRQL 값이 Dispatch 레벨이 되면 인터럽트 핸들러에서 등록된 핸들러들을 Dispatcher에서 수행하게 된다. 이와 같은 과정을 통해 RTiK-MP는 범용 운영체제인 윈도우에서 실시간 처리 기능을 제공하였다[3].

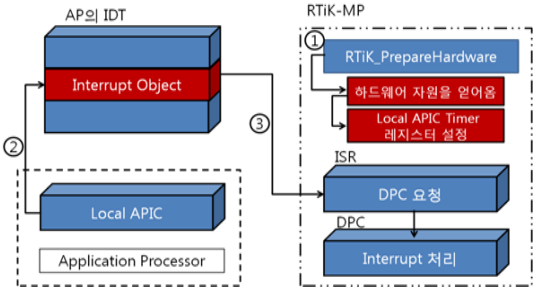

그림 2. RTiK-MP의 인터럽트 처리과정

2. 로컬 APIC 타이머 레지스터

윈도우에 실시간 처리 기능을 제공하기 위해 사용되는 RTX와 INtime, RTiK-MP는 로컬 APIC 타이머 관련 레지스터를 이용해 윈도우와는 독립적인 타이머 인터럽트를 발생시켜 실시간 쓰레드의 주기적인 동작을 보장한다. [그림 3]은 LVT(Local Vector Table)의 타이머 관련 레지스터에 관한 그림이다. LVT의 타이머 레지스터는 타이머의 상태정보를 유지하는 레지스터로써 타이머 인터럽트의 모드, 마스크 상태, 인터럽트 전달상태 및 인터럽트 벡터 넘버를 가진다. 초기 카운트 레지스터의 경우 타이머의 주기를 결정하는 레지스터로써, 설정된 카운트 값이 감소되어 0에 도달하면 타이머 인터럽트를 발생시키게 된다. 타이머 인터럽트가 발생하면 타이머 레지스터에 설정된 벡터를 통해 해당하는 IDT의 벡터 넘버로 분기하게 된다. IDT에 저장되어 있는 인터럽트의 핸들러 주소를 참조하여 주기적으로 발생되는 타이머 인터럽트의 핸들러를 수행한다. 하지만 윈도우 8의 경우 부팅 시 초기 카운트 레지스터와 현재 카운트 레지스터의 값을 0으로 초기화하기 때문에 정확한 인터럽트 발생을 위한 주기설정 값을 얻어 올 수 없는 문제점이 있다. 따라서 윈도우 8에 실시간 처리 기능을 제공하기 위해서는 초기 카운트 레지스터의 값을 설정하기 위한 연구가 필요하다[4-7].

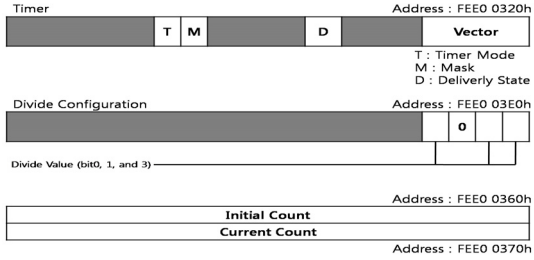

그림 3. 로컬 APIC 타이머 관련 레지스터

3. CPU에서 전력제어 현황

최근 휴대용 기기의 성능이 좋아지고 기능이 다양해짐에 따라 전력 관리에 대한 중요성이 부각되고 있다. 저전력 기법은 일정시간 동안 CPU의 이용률에 따라 CPU의 동작 상태를 제어함으로써 전력 소비량을 줄이

는 기법으로 배터리의 사용시간이 중요한 모바일 프로세서 및 플랫폼에서 중요성이 부각되고 있다. 일반적으로 x86 아키텍처에서의 휴대용 기기를 위해 저전력 기법으로 C-State라는 5가지의 CPU 상태를 제공하는데, [표 1]과 같다.

표에서 볼 수 있듯이 C0는 CPU가 프로세스들을 수행 시 동작하는 상태이며, C1과 C1E의 경우 CPU를 유휴상태로 만들어 클럭과 전압을 낮추어 전력 소비량을 줄여준다. C3와 C6의 경우 CPU의 동작을 멈추고 L1/L2 캐시의 동작도 멈춤으로써 저전력을 제공한다. 이는 윈도우에서 C-State의 상태에 따른 전력 감소를 위해 CPU의 클럭 및 전압을 이용함을 의미한다. 하지만 CPU 클럭의 변화는 로컬 APIC 타이머 인터럽트 처리에 대한 지연시간 발생 및 동기화의 문제로 인해 실시간 시스템에서 가장 중요한 시간 결정성에 영향을 미치는 문제점이 있다. 따라서 시간 결정성을 보장하기 위해 C-State를 효율적으로 제어하기 위한 연구가 필요하다[8][9].

표 1. CPU 전력 상태

상태	설명
C0	CPU 활성화 모드
C1	CPU Idle 모드
C1E	CPU 내부클럭 중지 및 전압 감소에 따른 Idle 모드
C3	모든 CPU 내/외부 클럭의 중지 및 L1/L2 캐쉬 비활성화
C6	CPU 상태 저장 후 CPU 전압을 차단한 Sleep상태

III. 태블릿 PC에 실시간 처리 기능 지원 방안

윈도우에 실시간 처리 기능 제공을 위해 개발된 기존 RTiK-MP의 경우 앞서 언급한 바와 같이 로컬 APIC 타이머 레지스터를 이용한다. 하지만 윈도우 8과 같은 모바일 운영체제의 경우 로컬 APIC의 타이머 레지스터인 초기 카운트 레지스터가 0으로 설정되어 윈도우와 독립적인 타이머 인터럽트 발생이 불가능해졌기 때문에 실시간 처리 기능을 제공하는데 문제점이 있다.

이러한 문제점을 해결하기 위해 본 논문에서는 x86 기반의 모바일 운영체제인 윈도우 8에 실시간 처리 기능을 제공하기 위해 윈도우와는 독립적인 타이머 인터럽트를 통한 실시간 태스크를 생성할 수 있도록 RTiK+를 설계하였다. 이를 위해 RTiK+를 디바이스 드라이버 형태로 구현하여 별개의 HAL을 가짐으로써, 각각의 프로세서가 가지고 있는 로컬 APIC 및 커널 자원에 접근할 수 있도록 구현하였다. [그림 4]는 RTiK+가 커널 자원의 접근을 통해 윈도우와는 독립적인 타이머 인터럽트를 발생시키고, 처리하기 위한 과정을 나타낸 그림이다. 그림에서 볼 수 있듯이 RTiK+는 윈도우에 실시간 처리 기능을 제공하기 위해 로컬 APIC와 커널 자원을 이용한다. 타이머 인터럽트를 발생시키기 전에 AP의 로컬 APIC 초기 카운트 값을 MSR_FSB_FREQ 레지스터를 이용하여 설정함으로써 윈도우와는 독립적인 타이머 인터럽트가 발생하도록 구현하였다. 인터럽트 처리를 위해 IDT에 인터럽트 오브젝트를 등록함으로써 실시간 태스크를 생성할 수 있도록 설계 및 구현 하였다. 또한 생성된 실시간 태스크의 시간 결정성을 보장해주기 위해 MSR_PKG_CST_CONFIG_CONTROL 레지스터를 이용하여 CPU의 동작모드를 제어함으로써 설정된 태스크 주기의 오차범위를 최소화시켜 주기성을 보장하도록 하였다[10-12].

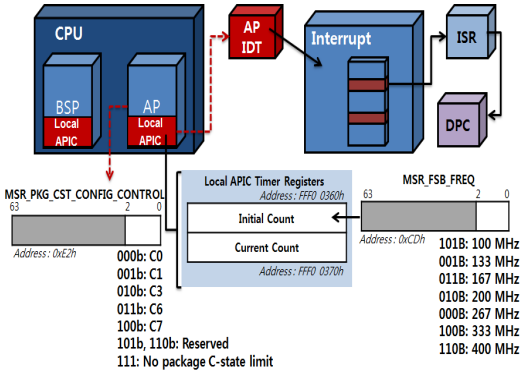

그림 4. RTiK+의 실시간 제공 방법

1. FSB 제어를 통한 타이머 인터럽트 발생

FSB(Front-Side-Bus)는 CPU와 메모리 컨트롤러 허브간의 데이터 통로를 의미하며, CPU가 주변 장치와 통신하기 위하여 사용되는 외부버스이다. 더불어 FSB

는 Multiplier를 이용하여 CPU 동작 속도를 결정짓는 기능을 제공하며, 메모리 클럭 속도에 영향을 주어 동기화를 제공한다. [그림 5]는 x86 아키텍처에서의 칩셋 블록 다이어그램을 나타낸 그림이다. 그림에서 볼 수 있듯이 클럭 발생기에서 생성된 주파수는 FSB에 연결되어 CPU에 제공되며, 이를 로컬 APIC 타이머 레지스터의 초기 카운트 값으로 설정되어 타이머 인터럽트의 발생 주기를 결정한다. 이는 CPU에서 발생되는 로컬 APIC 타이머가 FSB의 동작 주파수를 통해 제어됨을 의미하고, 모바일 운영체제인 윈도우 8의 경우 하드웨어 자원인 MSR_FSB_FREQ 레지스터의 하위 3비트를 통해 CPU에 따른 FSB 동작 주파수를 제공하고 있다. 따라서 RTiK+의 로컬 APIC 타이머 인터럽트 발생 주기를 결정하기 위해 MSR_FSB_FREQ 레지스터에 접근하여 FSB의 동작 주파수 값을 구하였고, 이를 로컬 APIC 타이머를 활성화하기 전에 초기 카운트 값으로 설정하여 윈도우와는 독립적인 타이머 인터럽트를 발생시키도록 하였다. 타이머 인터럽트가 발생하게 되면 인터럽트 핸들러로 분기하게 되고 수행되는 핸들러는 지연시간을 최소화하기 위해 지연처리함수를 호출하도록 설계 및 구현하였다[13].

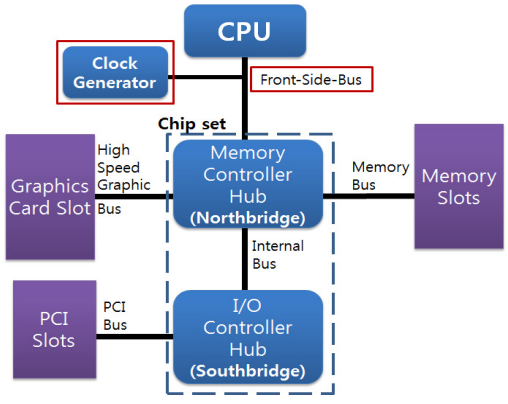

그림 5. x86 칩셋 블록 다이어그램

2. C-State 제어를 통한 시간 결정성 보장

윈도우는 CPU의 저전력을 위해 C-State라는 방식을 통해 CPU의 클럭을 낮추거나 전원을 차단함으로써, 전력 소비량을 줄여주는 실시간 절전 기능을 제공한다. 아래의 수식은 윈도우에서 CPU의 클럭을 결정하기 위해 사용하는 식으로써, FSB의 경우 시스템의 하드웨어에 따라 지정된 값을 가지고 있기 때문에 윈도우는 Multiplier 값의 제어를 통해 저전력을 제공함을 알 수 있다.

$$CPUClock = FSBFrequency * Multiplier \quad (1)$$

하지만 CPU의 클럭을 낮추기 위해 Multiplier 값을 조절하는 방식은 로컬 APIC의 타이머에 대한 시간 결정성을 보장하기 어려운 문제점이 있다. Multiplier 값의 빈번한 변동은 CPU의 로컬 APIC 타이머에 대한 처리속도를 변화시키고, 이는 이전에 설정된 초기 카운트 값을 통해 발생되는 주기적인 로컬 APIC 타이머 인터럽트와 시간적 오차를 발생시킨다. 더불어 Idle상태나 Sleep상태의 CPU를 활성화 시키는 시간으로 인해 지연 시간이 발생함으로 시간 결정성을 보장하지 못하는 문제점이 있다. 따라서 본 논문에서는 시간 결정성을 보장하기 위해 x86 아키텍처에서 제공하는 MSR_PKG_CST_CONFIG_CONTROL 레지스터에 접근하여 C-State를 제어하도록 설계 및 구현하였다.

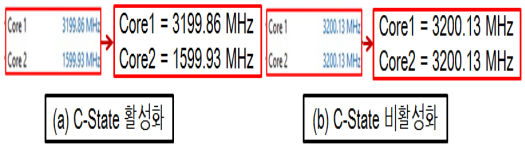

그림 6. C-State의 설정에 따른 CPU 클럭 변화

[그림 6]은 RTiK+를 이식 후 HWM BlackBox라는 프로그램을 이용하여 C-State의 설정에 따른 CPU의 클럭을 나타내었다. [그림 6]의 (a)에서 볼 수 있듯이 C-State가 활성화되면 CPU의 사용률에 따라서 클럭이 조절됨을 알 수 있고, (b)에서 볼 수 있듯이 비활성화 상태가 되면 사용률에 상관없이 클럭값을 최대값으로 유지한다. 이를 위해 RTiK+가 타이머 인터럽트를 설정하기 전에 커널 API인 *MmMapIoSpace()*를 사용하여 x86 아키텍처에서 제공하는 MSR_PKG_CST_CONFIG_CONTROL 레지스터에 접근하였고, 하위 3

비트의 값을 C0(000b)로 설정하여 CPU의 처리속도를 유지시킴으로써 RTiK+의 시간 결정성을 보장할 수 있도록 하였다. 더불어 RTiK+가 윈도우에서 제거되거나 비활성화 되었을 경우 이전상태로 변경되도록 구현함으로써 윈도우에 영향을 최소화하도록 설계하였다.

3. RTiK+의 사용자 영역의 실시간 처리 기능 제공

기존에 개발된 RTiK 계열의 경우 CPU의 로컬 APIC 타이머와 같은 커널 자원에 직접 접근하기 위해 디바이스 드라이버 형태로 구현되었다. 이를 통해 RTiK-MP의 경우 커널 영역의 하드웨어 자원에 접근이 용이하지만, 사용자 및 개발자가 커널 자원에 잘못된 접근 시 시스템에 악영향을 미칠 수 있는 문제점이 있다. 또한 기존의 연구했던 사용자 영역의 실시간 처리 기능을 위한 연구는 멀티코어 환경에서는 이벤트 시그널의 손실을 발생시켜 주기성 및 데이터의 정확성과 무결성을 보장하지 않는 문제점이 있다. 따라서 RTiK+는 데이터의 정확성과 무결성을 보장하기 위해 사용자 영역에서 쓰레드의 실시간 처리를 보장할 수 있도록 설계 및 구현하였다. [그림 7]은 RTiK+가 사용자 영역에 실시간 처리 기능을 제공하기 위한 동작과정을 나타낸 그림으로 사용자는 RTiK+가 제공하는 API를 이용하여 실시간 쓰레드를 생성하고, 생성된 실시간 쓰레드는 커널 영역으로부터 전달되는 주기적인 시그널을 받아 코드를 수행함으로써 쓰레드의 실시간 처리를 보장하도록 하였다[14].

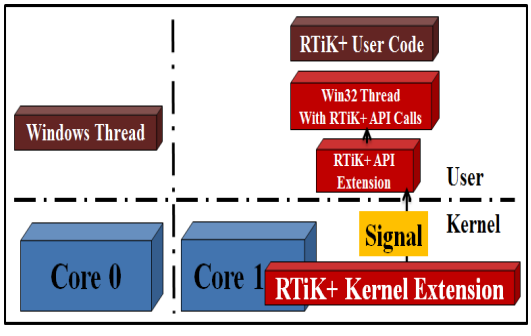

그림 7. RTiK+의 사용자 영역 동작과정

멀티코어 윈도우 환경의 경우 해당 코어에서 동작하는 프로세스에게만 이벤트 발생을 알리기 때문에 이벤트 시그널 손실이 발생하게 된다. 이를 해결하기 위해 생성된 실시간 쓰레드들을 RTiK+가 이식되어 있는 코어에서만 동작함으로써 이벤트 시그널 손실을 막았다. [그림 8]은 데이터의 정확성과 무결성을 보장하기 위해 RTiK+에서 구현한 코드이다. 그림에서 볼 수 있듯이 실시간 쓰레드 내부에서 *Process_Affinity()* 함수를 호출하도록 구현하였고, 쓰레드 내부에서 윈도우 API인 *SetProcessAffinityMask()* 함수 호출을 통해 RTiK+가 동작하는 코어에서만 실시간 쓰레드가 동작하도록 하였다. 또한 RTiK+의 실시간 쓰레드 우선순위와 프로세스 우선순위를 가장 높은 단계로 설정하여 생성함으로써, 다른 윈도우 쓰레드들에게 CPU를 선점 받지 않도록 하였다. [그림 9]는 RTiK+에서 사용자 영역에 실시간 처리를 제공하기 위해 구현한 코드이다. 그림에서 볼 수 있듯이 실시간 쓰레드 생성 시 윈도우에서 제공하는 가장 높은 쓰레드 우선순위인 THREAD_PRIORITY_TIME_CRITICAL과 가장 높은 태스크 우선순위인 REALTIME_PRIORITY_CLASS로 설정하여 윈도우의 다른 쓰레드들보다 먼저 동작하도록 구현하였다.

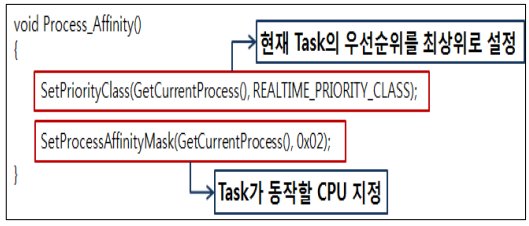

그림 8. 실시간 쓰레드가 동작할 코어 설정

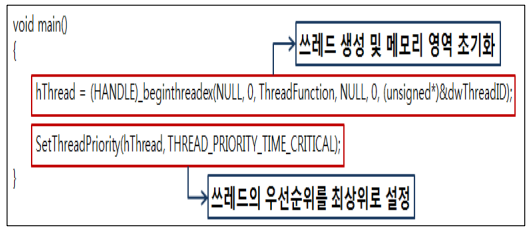

그림 9. 실시간 쓰레드의 우선순위 설정

Ⅳ. 실험환경 및 결과

1. 실험환경 및 실험방법

RTiK+가 모바일 운영체제인 윈도우 8에 정상적으로 이식되어 실시간 처리 기능을 제공함을 검증하기 위한 실험환경은 [그림 10]과 [표 2]와 같이 구성하였다. 호스트 컴퓨터에 RTiK+를 이식하였고, 호스트 컴퓨터와 타겟 컴퓨터를 USB-직렬 케이블로 연결하여 RTiK+를 디버깅 및 모니터링 하였다.

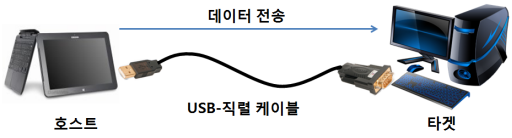

그림 10. 실험환경

표 2. 호스트 컴퓨터와 타겟 컴퓨터의 환경구성

	호스트	타겟
CPU	Intel Pentium Dual-Core 2117U @ 1.86 GHz	Intel Core i5-2500 @ 3.30 GHz
OS	Windows 8	Windows 7
Development tool	Visual Studio 2008	Visual Studio 2008

본 논문에서는 RTiK+의 성능을 확인하기 위해 클럭 틱의 수를 반환하는 RDTSC(Read Time Stamp Count) 명령어를 사용하여 주기를 측정하였다. [그림 11]은 실험을 위하여 실시간 쓰레드에서 구현한 코드를 나타낸 그림이다. 코드에서 볼 수 있듯이 실험 방법은 1ms의 주기로 설정된 타이머 인터럽트가 발생하여 사용자 영역의 실시간 쓰레드가 수행 될 때마다 클럭 틱 값을 배열에 저장하였고, RTiK+의 수행이 완료 되었을 때 저장해 두었던 클럭 틱 값의 차이를 계산하는 방법으로 주기측정을 하였다.

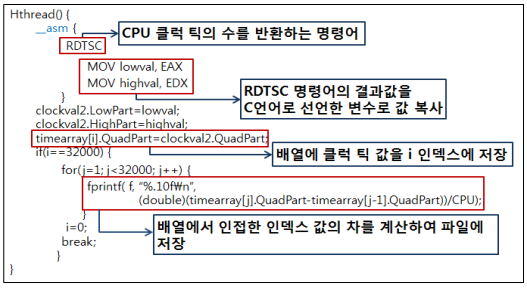

그림 11. 테스트 코드

또한 RTiK+에서 시간 결정성이 만족됨을 보이기 위해 동일한 환경에서 C-State의 상태에 따른 주기측정을 진행하였고, 실시간 쓰레드가 윈도우의 다른 쓰레드들에 영향을 받지 않고 CPU를 선점하여 동작함을 보이기 위해 무한으로 While문을 수행하는 워크로드를 생성하여 주기를 측정하였다.

2. 실험결과

[그림 12]과 [그림 13]은 C-State를 비활성화 시킨 후 RTiK+의 주기를 1ms로 설정하고, 워크로드가 없을 때와 10개의 워크로드를 RTiK+와 동시에 수행시켜 주기를 측정한 결과를 그래프로 나타낸 것이다. 엑셀에서는 그래프로 표현할 수 있는 데이터의 최대 개수가 32000개로 RTiK+의 동작을 32000번을 수행시켜 주기를 측정하였으며, x축의 값은 RTiK+의 동작횟수를 의미하고 y축은 주기 값을 의미한다.

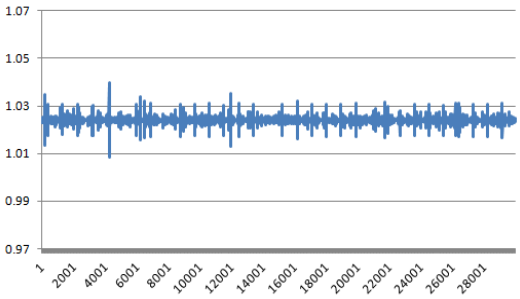

그림 12. RTiK+의 1ms 주기측정

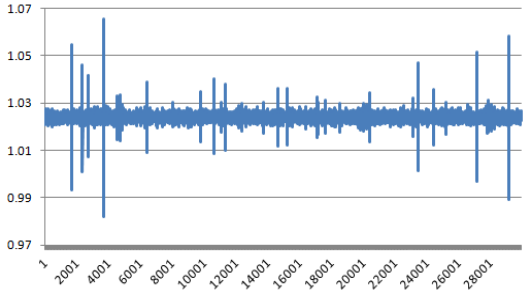

그림 13. 워크로드 10개와 RTiK+의 1ms 주기측정

[표 3]은 [그림 12]과 [그림 13]의 주기 측정결과의 최대값과 최소값을 나타낸 표이다. 표에서 볼 수 있듯이

실시간 쓰레드는 워크로드가 없을 경우 약 3%의 오차를 가지며 주기적으로 동작하는 것을 알 수 있다. 또한 10개의 워크로드를 적용했을 경우 약 6%의 오차를 가지며 주기적으로 동작함으로써 다수의 병렬 프로세스의 영향을 받지 않음을 알 수 있다.

표 3. RTiK+의 주기측정 및 워크로드 적용 시 주기측정

	RTiK+ 의 1ms 주기	워크로드 적용 1ms 주기
최대값	1.039777 ms	1.062404 ms
최소값	1.008465 ms	0.957421 ms

[표 4]는 동등한 환경에서 RTiK+의 주기를 변경하여 측정한 결과를 나타낸 표이다. 표에서 볼 수 있듯이 RTiK+의 주기를 3ms와 4ms, 5ms로 설정하여 테스트를 진행하였고, 각각의 경우 약 2%의 오차 범위 내에서 정상 동작함을 확인하였다.

표 4. RTiK+의 다양한 주기설정에 따른 측정결과

	RTiK+ 의 3ms 주기	RTiK+의 4ms 주기	RTiK+의 5ms 주기
최대값	3.066 ms	4.105 ms	5.118 ms
최소값	3.051 ms	4.094 ms	5.083 ms

[그림 14]는 C-State를 활성화시키고 RTiK+의 1ms의 주기를 가지는 실시간 쓰레드를 생성하여, 측정한 결과를 나타낸 그래프이다. 또한 [표 5]는 [그림 14]의 주기 측정결과의 최대값과 최소값을 나타낸 표이다. 표에서 볼 수 있듯이 C-State를 활성화시켰을 때 워크로드가 없는 경우에도 설정된 1ms 주기를 크게 벗어남을 알 수 있다. 이와 같이 C-State는 무기체계에서 중요한 시간 결정성을 보장하지 못함을 의미한다.

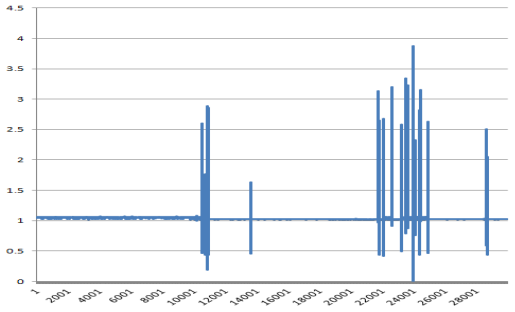

그림 14. C-State를 활성화 후 RTiK+의 주기측정

표 5. C-State를 활성화 후 주기측정

	C-State 활성화 후 주기 측정결과
최대값	3.876373 ms
최소값	0.001233 ms

V. 결론 및 향후 연구과제

본 논문에서는 윈도우 8의 태블릿 PC환경에 실시간 처리 기능을 제공하기 위해 MSR_FSB_FREQ 레지스터를 제어하여 윈도우와는 독립적인 타이머 인터럽트가 발생하도록 하였다. RTiK+의 시간 결정성을 보장해 주기 위해 MSR_PKG_CST_CONFIG_CONTROL 레지스터를 이용하여 CPU의 동작모드를 제어함으로써 설정된 주기로 정상 동작하도록 설계 및 구현하였다. 또한 사용자 영역에서 데이터의 정확성과 무결성을 보장하기 위해 실시간 쓰레드를 타이머 인터럽트가 발생되는 코어에서만 동작하도록 설계 및 구현하여, 실시간 처리 기능을 제공하도록 하였다.

구현된 RTiK+의 성능을 검증하기 위해 RDTSC를 사용하였고, 생성된 실시간 쓰레드의 최소주기인 1ms를 실험 했을 경우 워크로드가 없을 때 약 3%의 오차로 동작하며, 워크로드를 적용했을 경우 약 6%의 오차로 동작하는 것을 확인하였다. 또한 워크로드를 적용했을 시 빈번한 스케줄링 인터럽트가 발생해도 윈도우의 다른 쓰레드들에 영향을 받지 않고 CPU를 선점하여 동작하는 것을 증명하였다.

향후 연구과제로는 RTiK+의 높은 신뢰성을 검증하기 위해 실제 무기체계에서 사용되는 점검 장비에서의 테스트가 필요하고, 이를 바탕으로 개발자들이 RTiK+를 보다 편리하게 사용하기 위한 API 구현이 필요하다. 또한 앞으로 다양한 장비에 실시간 처리 기능을 제공하기 위하여 실시간 응용에 적용할 수 있는 확장 기능에 관한 연구진행이 필요하다.

참 고 문 헌

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

[2] http://www.tenasys.com
[3] 송창인, 이승훈, 주민규, 이철훈, “멀티프로세서 윈도우즈 상에서 실시간 처리 기능 지원”, 한국콘텐츠학회논문지, 제12권, 제6호, pp.68-77, 2012(6).
[4] Intel, *Intel 64 and IA-32 Architectures Software Developer's Manual Vol.1 : Basic Architecture*, Intel, 2012.
[5] Intel, *Intel 64 and IA-32 Architectures Software Developer's Manual Vol.3 : System Programming Guide*, Intel, 2012.
[6] Intel, *Intel 64 and IA-32 Architectures Software Developer's Manual Vol.2 : Instruction Set Reference,* Intel, 2012.
[7] Intel, *Intel 64 Architectures x2APIC Specification*, Intel, 2008.
[8] http://www.acpi.info/spec.htm
[9] Intel, *ADVANCED CONFIGURATION AND POWER INTERFACE SPECIFICATION Revision 5.0*, Intel, 2011.
[10] D.A. Godse and A.P. Godse, *Microprocessors*, Technical Publications Pune, 2007.
[11] O. Bailey, *Embedded systems : desktop integration*, Wordware Publishing, 2005.
[12] Intel, *MultiProcessor Specification Version 1.4*, Intel, 1997.
[13] http://en.wikipedia.org/wiki/Front-side_bus
[14] 주민규, 이진욱, 김종진, 조한무, 박영수, 이철훈, “x86 기반의 윈도우즈 상에서 실시간 처리 기능 지원 방법”, 한국차세대컴퓨팅학회논문지, 제11권, 제4호, pp.47-58, 2011(8).

저 자 소 개

박 지 윤(Ji-Yoon Park) 준회원

- 2013년 2월 : 충남대학교 컴퓨터공학과(공학사)
- 2013년 2월 ~ 현재 : 충남대학교 컴퓨터공학과 석사과정 재학

<관심분야> : 실시간 운영체제, 임베디드 시스템, 실시간 통신

조 아 라(Ah-Ra Jo) 준회원

- 2012년 2월 : 충남대학교 컴퓨터공학과(공학사)
- 2012년 2월 ~ 현재 : 충남대학교 컴퓨터공학과 석사과정 재학

<관심분야> : 실시간 운영체제, 임베디드 시스템, 실시간 통신

김 효 중(Hyo-Joung Kim) 준회원

- 2007년 2월 : 충남대학교 컴퓨터공학과(공학사)
- 2009년 2월 : 충남대학교 정보통신공학과(공학석사)
- 2012년 12월 ~ 현재 : LIG넥스원 선임연구원

<관심분야> : 유도무기, 점검장비, 임베디드 시스템

최 정 현(Jung-Hyun Choi) 정회원

- 2000년 2월 : 충남대학교 컴퓨터과학과(이학사)
- 2002년 2월 : 충남대학교 컴퓨터과학과(이학석사)
- 2005년 2월 : 충남대학교 컴퓨터과학과(이학박사)
- 2008년 7월 ~ 현재 : LIG넥스원

<관심분야> : 유도무기, 점검장비, 실시간 운영체제, 임베디드 시스템

허 용 관(Yong-Kwan Heo) 정회원

- 1991년 2월 : 한양대학교 전자계산학과
- 2006년 10월 ~ 현재 : LIG넥스원

<관심분야> : 유도무기, 실시간 통신, 임베디드 시스템

조 한 무(Han-Moo Jo) 정회원

- 1995년 : 숭실대학교(공학사)
- 1994년 ~ 현재 : LIG넥스원 책임연구원

<관심분야> : 유도무기, 점검장비, 임베디드 시스템

이 철 훈(Cheol-Hoon Lee) 정회원

- 1983년 2월 : 서울대학교 전자공학과(공학사)
- 1988년 2월 : 한국과학기술원 전기 및 전자공학(공학석사)
- 1992년 2월 : 한국과학기술원 전기 및 전자공학과(공학박사)
- 1983년 3월 ~ 1986년 2월 : 삼성전자 컴퓨터 사업부 연구원
- 1992년 3월 ~ 1994년 2월 : 삼성전자 컴퓨터 사업부 선임연구원
- 1994년 2월 ~ 1995년 2월 : Univ. of Michigan 객원연구원
- 1995년 5월 ~ 현재 : 충남대학교 컴퓨터공학과 교수
- 2004년 2월 ~ 2005년 2월 : Univ. of Michigan 초빙연구원

<관심분야> : 실시간시스템, 운영체제, 고장허용 컴퓨팅, 로봇 미들웨어