실시간 운영체제 iRTOS상에서의 HISR을 이용한 DPC설계 및 구현

Design and Implementation of DPC using a HISR on iRTOS Real-Time Operating System

권재국, 손재열*, 이철훈
충남대학교 컴퓨터공학과, (주)한화 종합 연구소*

Jae-Guk Gwon , Jae-Yeol Son *, Cheol-Hoon Lee
Chungnam National Univ., Hanwha R&D Center.*

요약

실시간 운영 체제는 운영체제로서 논리적 정확성이 중요하지만 시간적 정확성 또한 중요한 운영체제이다. 그렇기 때문에 시스템에서 발생하는 사건들을 처리할 때의 지연시간은 낮아야 한다. 특히 인터럽트 서비스 루틴에서는 현재 인터럽트 레벨보다 낮은 인터럽트는 마스킹되기 때문에 모든 인터럽트들이 원활히 동작하기 위해서 인터럽트 서비스 루틴은 보다 짧은 시간 동안만 CPU를 점유해야 한다. 처리시간이 긴 인터럽트 서비스 루틴의 지연시간을 줄이기 위해 윈도우 운영체제에서는 DPC(Deferred Procedure Call)를 이용하고 있다. 본 논문에서는 실시간 운영체제 iRTOS상에서 HISR을 이용하여 DPC를 설계 및 구현 하였다.

I. 서론

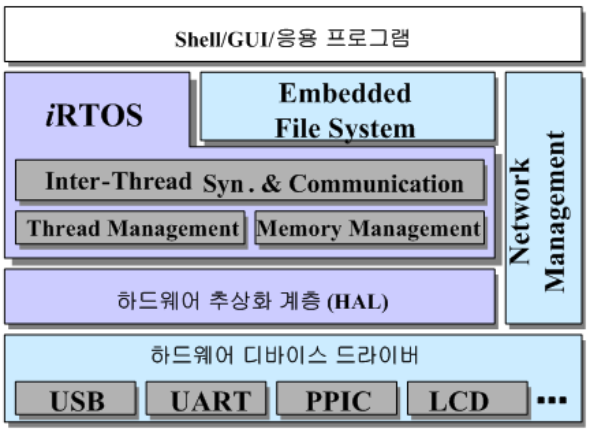
운영 체제는 시스템에서 발생하는 사건을 인터럽트에 의해 처리하는데, 이것은 비동기적인 사건의 발생을 CPU에게 알리는 하드웨어 메커니즘으로서 인터럽트 서비스 루틴(Interrupt Service Routine : ISR)을 통해 인터럽트에서 처리해야 할 부분을 수행한다. 인터럽트가 발생하면 CPU는 수행중이던 쓰레드를 멈추고 인터럽트 서비스 루틴으로 CPU점유권을 넘긴다. 인터럽트 서비스 루틴에서는 현재 인터럽트 레벨보다 낮은 인터럽트는 마스킹되기 때문에 시간 결정성을 저해시킨다. 이를 해결하기 위해 인터럽트 서비스 루틴은 보다 짧은 시간 동안만 CPU를 점유해야 하고 처리시간이 긴 인터럽트 서비스 루틴은 지연처리 호출 (Deferred Procedure Call : DPC)에서 처리하여 인터럽트 지연시간을 줄일 필요성이 있다. 이에 본 논문은 실시간 운영체제 iRTOS상에서 HISR(High Level Interrupt Service Routine)을 이용하여 윈도우 운영체제의 메커니즘인 지연처리호출을 설계 및 구현 하였다.

본 논문은 2장에서 관련연구로서 실시간 운영체제 iRTOS와 HISR, 지연처리호출에 대하여 설명하고, 3장에서는 HISR을 이용한 지연처리호출의 설계 및 구현을, 4장에서는 테스트 환경과 결과를 보이고 마지막으로 5장에서는 결론 및 향후 연구과제를 기술한다.

Ⅱ. 관련연구

2.1 iRTOS

실시간 운영체제 iRTOS는 멀티테스킹과 0~255 까지 총 256단계의 우선순위 기반의 스케줄링을 제공한다. 그리고 효율적인 자원관리 및 쓰레드간 통신을 위해 세마포어, 메시지 큐를 제공 한다.



▶▶ 그림 1. iRTOS 전체 구성도

2.2 HISR

HISR이란 인터럽트 서비스 루틴기능을 인터럽트 레벨이 아닌 쓰레드보다는 약간 높은 우선순위 레벨에서 수행하는 것을 말한다. HISR은 동적으로 생성되고 삭제될 수 있고, 이를 위해 HISR마다 자신의 스택을 가지며 쓰레드와 비슷한 제어 블록을 가지고 있다. HISR은 0~2 의 우선순위를 부여해서 활성화된 HISR이 있으면 항상 쓰레드보다 먼저 CPU를 점유한다[1].

2.2.1 HISR 생성과 삭제

표 1. HISR생성과 삭제 함수

API 함수	설명
CreateHISR()	HISR을 생성하는 함수
DeleteHISR()	HISR을 삭제하는 함수

HISR을 생성하기 위해서 CreateHISR()함수를 사용하고 함수의 인자로는 HISR의 제어 블록, 이름, 실행 함수 포인터, 우선순위, 스택 포인터 및 스택 크기이다. HISR을 삭제하기 위해서 DeleteHISR()함수를 사용하고 함수의 인자로는 HISR의 제어블록이다.

2.2.2 HISR 제어

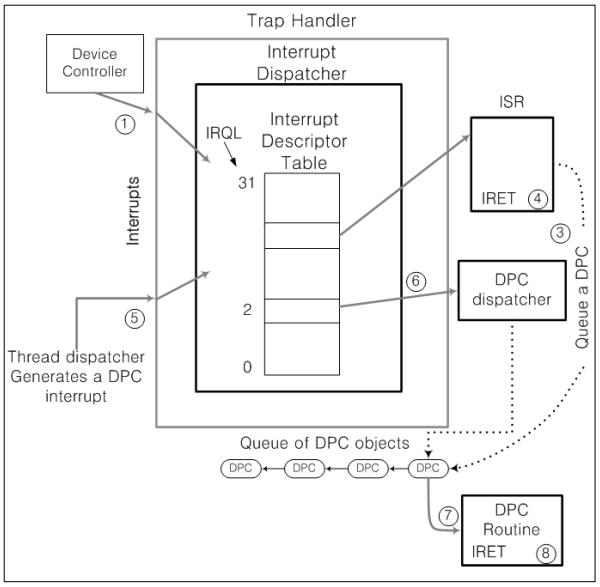
표 2. HISR 제어 함수

API 함수	설명
ActivateHISR()	HISR을 활성화 시키는 함수
GetCurrentHISR()	현재 실행중인 HISR을 얻는 함수

HISR을 ActivateHISR()함수를 사용하여 활성화 시킨다. 함수의 인자는 HISR의 제어 블록이다. 활성화된 HISR은 다른 쓰레드보다 우선순위가 높기 때문에 먼저 CPU를 점유하게된다.

2.3 지연처리호출

지연처리호출은 하드웨어 인터럽트 서비스를 적절하게 제공하기 위해 윈도우에서 제공하는 서비스이다. 인터럽트 서비스 루틴중 덜 중요하거나 우선 처리 하지 않아도 되는 부분을 지연처리호출에서 처리한다.



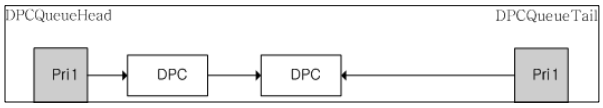
▶▶ 그림 2. DPC 수행 과정

[그림 2]는 지연처리호출의 수행 과정을 나타내고 있다. 인터럽트가 발생하고 이때 발생한 인터럽트의 IRQL(Interrupt Request Level)이 높은 경우 현재의 문맥을 저장하고 IDT (Interrupt Descriptor Table)을 참조하여 인터럽트 서비스 루틴을 실행한다. 인터럽트 서비스 루틴 수행시 IRQL은 높아지고 인터럽트 서비스 루틴중에서 덜 중요한 부분을 지연처리호출큐에 삽입한다. 인터럽트 서비스 루틴이 종료되면 IRQL은 낮아지며 지연처리호출큐 상의 인터럽트들을 실행하고 큐의 모든 오브젝트를 완료하면 원래 쓰레드로 복귀한다[2].

Ⅲ. HISR을 이용한 지연처리호출 설계 및 구현

3.1 지연처리호출큐

생성되는 지연처리호출 오브젝트들은 이중 순환 연결 리스트로 관리 하도록 구성하였으며, 새로 생성될 경우 리스트의 가장 끝에 삽입된다. 생성을 마친 지연처리호출 오브젝트는 지연처리호출큐에 삽입하지 않는다. 지연처리호출큐는 HISR의 Priority가 1인 Activate List를 이용하여 관리하였다. Activate List를 지연처리호출 용도로 사용하기 위해 별도의 DPCQueueHead 와 DPCQueueTail을 구현 하였다. 함수를 통해 지연처리호출큐의 마지막에 삽입하며, 지연처리호출큐에 먼저 삽입된 지연처리호출 오브젝트를 우선으로 스케줄링 한다.



▶▶ 그림 3. 지연처리호출큐

3.2 지연처리호출생성

표 3. 지연처리호출생성 함수

API 함수	설명
CreateDpcRequest()	지연처리호출을 생성하는 함수

위 함수는 HISR의 CreateHISR()함수를 호출 하며 지연처리호출에게 HISR제어블록 한 개를 제공한다. 그리고 제공한 HISR제어블록의 디스크립터 번호를 리턴하게 된다. 생성된 지연처리호출은 전체 지연처리호출 리스트에 삽입만 될 뿐 스케줄링 되지 않는다.

3.3 지연처리호출제어

표 4. 지연처리호출큐 삽입 함수

API 함수	설명
ActiveDpcRequest()	지연처리호출을 큐에 삽입하는 함수

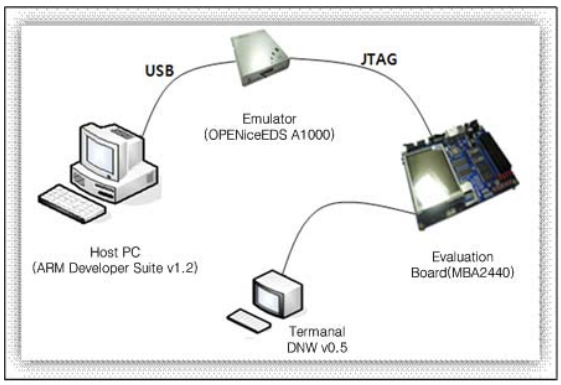
위 함수는 HISR의 ActivateHISR()을 호출 하며, 지연처리호출을 생성시 제공 받은 HISR제어블록을 이용하여 지연처리호출 오브젝트를 수행하기 때문에 이 함수를 호출하기 이전에 CreateDpcRequest() 함수를 호출하여 지연처리호출 생성을 마치고 HISR제어블록을 제공 받은 후에 사용해야 한다. 지연처리호출큐에 들어간 지연처리호출 오브젝트들은 다른 쓰레드보다 먼저 스케줄링 되어 CPU를 점유한다.

3.4 지연처리호출스케줄링

iRTOS의 스케줄러는 쓰레드를 스케줄 하기 전에 지연처리호출큐에 지연처리호출 오브젝트존재 여부를 확인한다. 지연처리호출이 존재하면 수행중이던 현재 쓰레드와 지연처리호출 오브젝트의 수행함수의 문맥교환이 발생하여 지연처리호출이 수행된다. 이때 다른 쓰레드보다 지연처리호출 오브젝트들이 먼저 CPU를 점유하게 되며 지연처리호출큐 내의 오브젝트모두가 스케줄링이 되어야 다른 쓰레드가 CPU를 점유 할 수 있다. 그리고 지연처리호출 오브젝트의 함수를 수행 할 동안에는 인터럽트발생 가능 상태이기 때문에 인터럽트들에 대하여 응답할 수 있다.

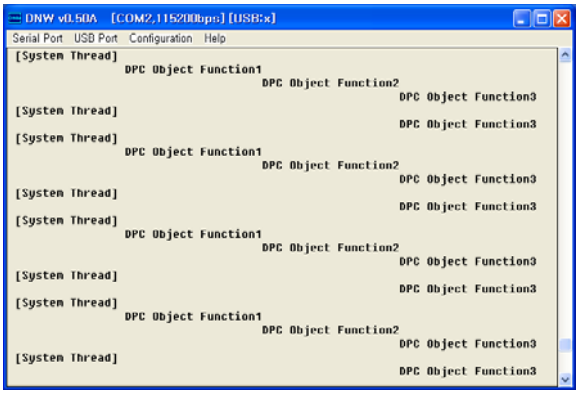
Ⅳ. 테스트 및 결과

본 논문에서 구현한 지연처리호출을 ARM-920T 기반의 S3C2440 MCU가 탑재된 MBA2440보드에서 수행하였다. 개발 도구로는 ARM Developer Suit v1.2, 디버거로는 OPENiceEDS-A1000 에뮬레이터를 사용하였다.



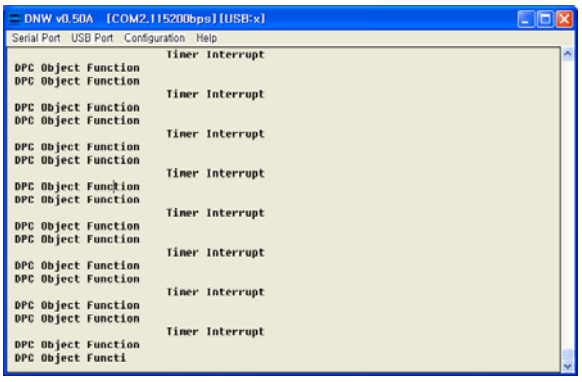
▶▶ 그림 4. 개발환경

[그림 5]는 리얼타임 쓰레드와 지연처리호출이 스케줄링 되는 결과 화면이다. "[System Thread]"는 시스템에서 동작하고 있는 리얼타임 쓰레드이다. DPC Object1, DPC Object2는 5 클럭틱 주기마다, DPC Object3은 2 클럭틱 주기마다 ActiveDpcRequest() 함수로 지연처리 호출큐에 삽입 하였다.



▶▶ 그림 5. DPC생성과 활성화

테스트 결과 기존에 수행 중이던 "System Thread" 보다 지연처리호출큐에 들어온 DPC Object 1, 2, 3 이 CPU를 점유한 모습을 볼 수 있다. 그리고 DPC Object는 FIFO(First In First Out)로 CPU를 점유 하였다. [그림 6]은 지연처리호출 오브젝트의 수행 함수가 동작하고 있을때 타이머 인터럽트가 발생하는 결과이다.



▶▶ 그림 6. 지연처리호출 수행 함수와 인터럽트

V. 결론 및 향후 연구 과제

본 논문에서는 실시간 운영체제 iRTOS상에서 인터럽트를 처리하기 위한 인터럽트 서비스 루틴 일부분을 지연처리호출로 스케줄링이 가능하게 구현하여 인터럽트 지연시간을 단축하였고, 이로 인해 시스템의 사건에 대한 응답성 늦어지는 문제를 어느 정도 해결할 수 있었다. 향후에는 지연처리 호출사용 여부에 따라 인터럽트 지연시간이 어느 정도 개선되는지 측정 하는 것이 목표이다.

▌참 고 문 헌▐

[1] 이재규 외 3명, "실시간 운영체제를 위한 2단계 인터럽트 서비스 루틴의 설계 및 구현" 한국정보과학회, 학술발표논문집 제31권 제1호, pp. 160~162, 2004

[2] DPC : http://recipes.egloos.com/5000239

[3] David Setpner 외 2명, "Embedded Application Design Using a Real-Time OS", IEEE 1999

[4] Jane W.S. Liu, "RealTimeSystems," PrenticeHall, 2000

[5] 박윤미 회 3명, "실시간 운영체제를 위한 태스크 스케줄링의 설계 및 구현" 한국정보과학회, 학술발표논문집 제30권 제2호, pp.298~300, 2003