MPC750 프로세서 기반의 내장형 시스템을 위한 실시간 운영체제 설계 및 구현

박윤미[0], 이득영[†], 김도훈[†], 이철훈
충남대학교 컴퓨터공학과, [†]삼성탈레스㈜ R&D팀
(ympark[0], chlee)@.ce.cnu.ac.kr, (dukyung.lee[†], dh71.kim[†])@samsung.com

Design and Implementation of A Real-Time Operating System for Embedded System based on MPC750 Processor

Yoon-Mi Park[0], Duk-Yung Lee[†], Do-Hoon Kim[†], and Cheol-Hoon Lee
Dept. of Computer Engineering, Chungnam National Univ.
[†] Dept. of R&D Team, SAMSUNG THALES CO.,LTD

요 약

실시간 운영체제는 그 특성상 범용 운영체제와는 달리 시간 결정성(determinism)을 보장하는 안정된 스케줄링 기능을 갖춘 운영체제이다. 현재 실시간 운영체제를 필요로 하는 내장형 시스템들은 비싼 사용료를 지불하며 외국의 상용 실시간 운영체제를 도입하여 제품 개발에 활용하고 있다. 상용 실시간 운영체제를 사용할 경우, 운영체제 자체는 블랙 박스(바이너리 소스)이기 때문에 세밀한 제어가 불가능하고 불필요한 기능들을 포함하고 있다. 그러므로 독자적인 운영체제 개발 및 확보가 중요하다. 본 논문은 MPC750 프로세서에 기반한 실시간 운영체제를 개발함에 목적이 있다.

1. 서론

실시간 운영체제는 그 특성상 범용 운영체제와는 달리 시간 결정성(determinism)을 보장하는 안정된 스케줄링 기능을 갖춘 운영체제이다. 국외에서 상용 실시간 운영체제가 활발하게 연구되고 있는 것에 반하여, 국내에서는 아직 초기 단계에 있기 때문에 현재 실시간 운영체제를 필요로 하는 내장형 시스템들은 비싼 사용료를 지불하여 외국의 상용 실시간 운영체제를 제품 개발에 활용하고 있다. 업체가 제공하는 운영체제 자체는 블랙 박스(바이너리 소스)이기 때문에 세밀한 제어가 불가능하고 불필요한 기능들을 포함하고 있다. 지금까지 많은 방산 분야에서 초창기의 콘트롤러 정도의 운영체제를 사용하거나 실제로는 사용하지 않는 기능의 운영체제를 구매하여 여러 가지 불편과 제약을 감수하면서 사용해 오고 있다. 그러므로 독자적인 운영체제 개발 및 확보가 중요하다. 본 논문은 MPC750 프로세서에 기반한 독자적인 실시간 운영체제를 개발하여 경제적인 부담을 덜어주고, 내장형 시스템의 응용프로그램 요구에 맞게 필요한 모듈만을 선택하여 적용함으로써 적은 메모리를 소모하면서도 성능을 향상시키는데 목표가 있다.

본 논문의 2 장에서는 관련 연구를, 3 장에서 실시간 운영체제 설계 및 구현을, 4 장에서는 테스트 환경 및 결과, 5 장에서는 결론 및 향후 연구 과제를 기술한다.

2. 관련 연구

2.1 MPC750

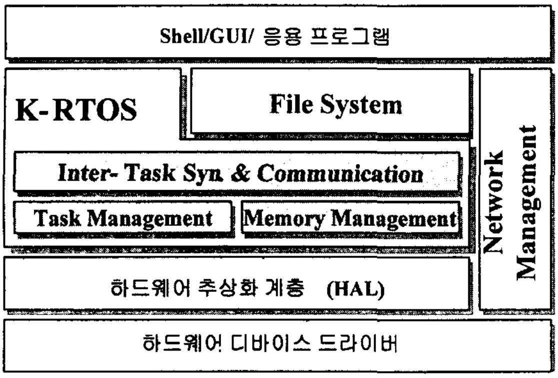
MPC750 은 산업제어 및 정보 통신 장비의 임베디드 분야의 고성능 기기에 적용되고 있는 프로세서 이다. MPC750 프로세서는 MMU(Memory Management Unit)를 사용할 수 있으며, 데이터 캐시와 명령어 캐시를 통합하고 있는 32bit PowerPC 이다. MPC750 의 프로그래밍 모델은 사용자 프로그래밍 모델과 슈퍼바이저 프로그래밍 모델 두 부분으로 나누어진다. 각 프로그래밍 모델은 32 개의 GPR(General Purpose Register), 32 개의 FPR(Floating-Point Register), 그리고 SPR(Special Purpose Register) 레지스터를 가진다. 단, 프로세서의 상태, 주소 번역 메커니즘, supervisor 레지스터를 제어하는 명령어는 슈퍼바이저 프로그래밍 모델에서만 제어된다.

2.2 실시간 운영체제

실시간 시스템은 시스템의 내·외부에서 발생하는 이벤트에 대하여, 이벤트 발생 시점에서 그 이벤트 처리가 끝날 때까지의 지연 시간이 미리 제시된 시간(Deadline)을 넘지 않도록 처리하는 시스템을 의미한다. 즉, 이벤트 발생과 처리가 실시간으로 이루어지는 시스템을 의미한다.

[그림 1]은 실시간 시스템의 전체 구성도를 보여 주고 있다. 가장 하위 단계에 하드웨어와 하드웨어의 초기화 및 관리를 담당하는 하드웨어 디바이스 드라이버가 존재하며, K-RTOS 커널과 하드웨어 사이의 인터페이스를 제공하는 하드웨어 추상화 계층(Hardware Abstraction Layer)이 그 위에 위치하며, K-RTOS 커널과 Embedded File System, Network Management 가 있으며, 최 상위 계층에는 Shell, GUI, 응용 프로그램(여러 개의 태스크)이 있다.

본 논문에서 개발된 실시간 운영체제인 K-RTOS 는 우선순위 기반의 선점형 커널이고, 실시간 운영체제의 핵심이라고 할 수 있는 멀티 태스킹(MultiTasking) 및 ITC 환경을 제공한다.



[그림 1] 실시간 시스템 전체 구성도

3. 실시간 운영체제 설계 및 구현

3.1 스케줄링 정책

K-RTOS 실시간 운영체제의 스케줄러는 우선순위 기반의 선점형 스케줄링을 하고 동일한 우선순위에 대해서는 기본적으로 라운드 로빈(Round Robin)을 스케줄링하며 필요에 따라 FIFO 스케줄링도 지원한다. 태스크의 우선순위는 0부터 63까지 64단계의 우선순위를 가지며 각각의 우선순위마다 *Ready List*가 존재한다. 스케줄러는 *Ready List*에서 가장 높은 우선순위를 찾아 실행한다.

3.2 태스크의 상태 전이도

각각의 태스크는 SUSPEND, READY, RUNNING, DELAYED, PENDING, 5 가지 상태 중 하나의 상태를 가지며, 특정 자원을 획득하기 위해 대기할 때 타임 아웃을 양수 값(Positive Value)으로 설정할 경우 PENDING+ DELAYED 상태가 된다.

3.3 ITC(Inter-Task Communication)

ITC 환경은 태스크와 태스크의 공동 작업(Cooperation)을 위한 동기화(Synchronization) 및 통신(Communication)을 위하여 세마포, 메시지 메일 박스, 메시지 큐, 메시지 포트 등을 지원하고 있다.

- 세마포(Semaphore)
  공유 자원의 안전한 관리를 위한 상호베제(Mutual Exclusion)와 태스크간 동기화에 사용된다.
- 메시지 큐(Message Queue)
  특정 태스크나 ISR 에서 다른 태스크로 여러 개의 메시지를 전달할 때 사용된다.
- 메시지 메일 박스(Message Mailbox)
  메시지 큐의 특수한 경우로 하나의 메시지를 빠르게 전송할 때 사용된다.
- 메시지 포트(Message Port)
  메시지 큐와 같이 여러 개의 메시지를 다른 태스크로 전달할 때 사용한다. 메시지 큐는 메시지를 복사해서 전달하지만 메시지 포트는 메시지의 포인터만을 전달한다.

3.4 메모리 관리 체계

K-RTOS 커널은 동적 메모리 관리를 위해 두 단계의 메모리 관리 기법을 제공한다. 하위 단계는 가변 크기의 메모리를 할당 또는 해제할 수 있는 힙 스토리지 매니저(Heap Storage Manager)이고 상위 단계는 고정 크기의 메모리를 할당 및 해제할 수 있는 메모리 풀(Memory Pool)이다.

3.5 문맥 교환

실시간 운영체제는 태스크의 중요도에 따라 우선순위를 부여 받으며, 시간 결정성 보장을 위해서 실행 중인 태스크보다 더 높은 우선순위의 태스크가 READY 상태가 되거나, ISR 수행 후에 문맥 교환이 일어난다. 문맥 교환은 CPU를 점유하고 있던 낮은 우선순위의 태스크는 수행이 중단되고, 현 시점의 PC와 CPU 레지스터들을 자신의 스택 영역에 저장하고 CPU를 점유할 태스크의 문맥을 자신의 스택에서 읽어와서 수행을 계속한다. 문맥 교환은 프로세서에 따라 CPU 레지스터의 구성이 달라지므로 커널과는 독립적이다.

[표 1]는 MPC750에서 교환되어야 할 문맥으로, 모두 39개 레지스터의 값으로 이루어진다.

[표 1] MPC750의 문맥

레지스터	기 능
R0 ~ R31	General Purpose Register
SRR0, SRR1	Used during exception processing
CTR	Count Register
XER	Integer exception Register
CR	Condition Register
LR	Link Register (Return address)
MSR	Machine State Register

3.6 예외 처리(Exception Handling)

K-RTOS 커널은 프로세서에 의존적인 외부 인터럽트를 포함하여 시스템 콜과 같은 여러 가지 예외 상황에 대한 처리를 한다. 이때 동작 모드는 Supervisor 모드이다. [표 2]는 예외 벡터를 나타내고 있고, 각 벡터는 [표 2]에서 보는 바와 같이 Address가 가리키는 물리 메모리 주소에 정확히 위치되어야 한다.

[표 2] Exception Vector Address

Exception	Vector offset
System reset	0x0100
Machine check	0x0200
DSI	0x0300
ISI	0x0400
External interrupt	0x0500
Alignment	0x0600
Program	0x0700
Floating-point unavailable	0x0800
Decrementer	0x0900
System call	0x0C00
Trace	0x0D00

MPC750 은 PowerPC 기반으로 예외 상황에 대한 핸들러가 있을 곳의 위치를 고정시키는 방식을 사용하며, [그림 2]는 이 방식을 적용하여 각 예외 처리에 대한 처리 루틴을 등록하는 코드이다.

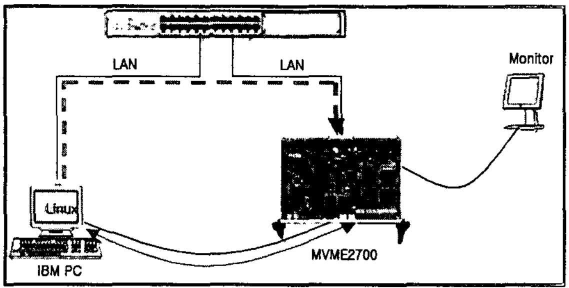
```
.= 0x0100
B MK-ResetHandler
.= 0x0200
B MK-MachinechkHandler
.= 0x0300
B DSIHandler

.......................................
```

[그림 2] Exception Vector 등록

4. 테스트 환경 및 결과

본 논문에서는 MPC750 CPU가 탑재된 MVME2700 Board을 대상으로 진행되었다. [그림 3]과 같이 Linux환경에서 크로스 컴파일러를 사용하여 실시간 운영체제 소스 코드 컴파일 및 디버깅을 수행하였다. 컴파일된 실행 이미지를 다운로드하기 위하여 TFTP를 설정하고, 이더넷을 통하여 다운로드되고, 실행 결과는 UART를 통해 모니터에 출력된다.

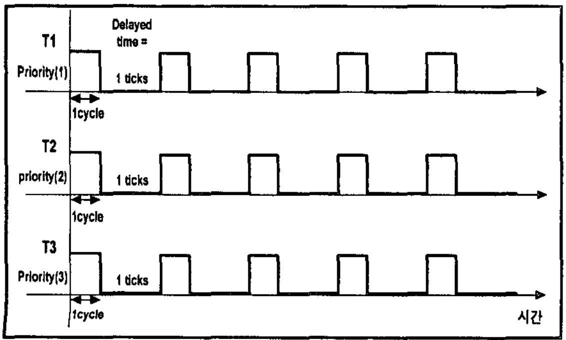


[그림 3] 개발 환경

본 논문에서 개발한 실시간 운영체제가 안정적으로 커널 서비스를 수행하는지 확인하기 위해, 멀티 태스킹 동작을 중점적으로 테스트하였다.

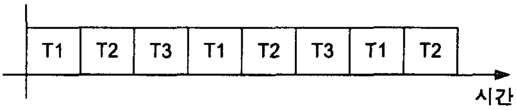
- 멀티 태스킹

[그림 4]는 서로 다른 우선순위를 가지는 각 태스크들의 유형을 나타낸 것이다.



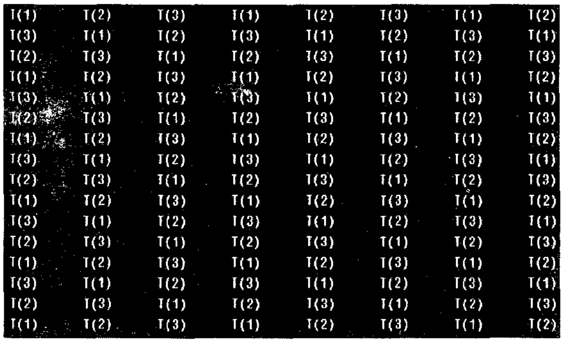
[그림 4] 멀티 태스킹 - 각 태스크의 유형

[그림 4]에서 보는 바와 같이 태스크들이 수행되었을 때, 실행 상태를 예측하면 [그림 5]와 같다.



[그림 5] 태스크들의 실행 상태

[그림 6]는 실제 동작 결과를 나타낸 것이다.



```
T(1)  T(2)  T(3)  T(1)  T(2)  T(3)  T(1)  T(2)
T(3)  T(1)  T(2)  T(3)  T(1)  T(2)  T(3)  T(1)
T(2)  T(3)  T(1)  T(2)  T(3)  T(1)  T(2)  T(3)
T(1)  T(2)  T(3)  T(1)  T(2)  T(3)  T(1)  T(2)
T(3)  T(1)  T(2)  T(3)  T(1)  T(2)  T(3)  T(1)
T(2)  T(3)  T(1)  T(2)  T(3)  T(1)  T(2)  T(3)
T(1)  T(2)  T(3)  T(1)  T(2)  T(3)  T(1)  T(2)
T(3)  T(1)  T(2)  T(3)  T(1)  T(2)  T(3)  T(1)
T(2)  T(3)  T(1)  T(2)  T(3)  T(1)  T(2)  T(3)
T(1)  T(2)  T(3)  T(1)  T(2)  T(3)  T(1)  T(2)
T(3)  T(1)  T(2)  T(3)  T(1)  T(2)  T(3)  T(1)
T(2)  T(3)  T(1)  T(2)  T(3)  T(1)  T(2)  T(3)
T(1)  T(2)  T(3)  T(1)  T(2)  T(3)  T(1)  T(2)
T(3)  T(1)  T(2)  T(3)  T(1)  T(2)  T(3)  T(1)
T(2)  T(3)  T(1)  T(2)  T(3)  T(1)  T(2)  T(3)
T(1)  T(2)  T(3)  T(1)  T(2)  T(3)  T(1)  T(2)
```

[그림 6] 실행 결과

5. 결론 및 향후 연구 과제

본 논문은 MPC750 을 위한 실시간 운영체제를 설계하고 구현 하였다. 본 논문에서 개발한 실시간 운영체제는 기본적으로 멀티 태스킹을 위한 커널 자료구조와 태스크 관리 및 스케줄링, 태스크간 통신 및 동기화 등의 서비스를 제공하고 있으며, 부가적인 기능으로 동적 메모리 관리, 타이머, Signal 등을 모듈별로 지원한다. 내장형 시스템은 응용프로그램의 요구에 맞게 필요한 모듈만을 선택하여 시스템에 적재함으로써 적은 메모리를 소모하면서도 성능 향상을 가질 수 있도록 구현하였다.

향후 연구 과제로는 미국의 POSIX 와 일본의 μ ITRON 과 같이 국가적 차원의 표준 인터페이스를 지원할 수 있도록, 국내에서 개발되는 실시간 운영체제들과 표준화를 위한 노력이 진행되어야 할 것이다.

6. 참고 문헌

[1] Jean, J. Labrosse, “ uC*/OS II The Real-Time Kernel*” , R&D Publications, 1999.

[2] MOTOROLA, “ *MPC750 RISC MicroprocessorFamily User’ s Manual*” , 2001

[3] 박희상, 정명조, 조희남, 이철훈, “ *Design of Open Architecture Real-Time OS Kernel*” , 한국정보과학회, 2002

[4] MOTOROLA, “ *(MVME2700 Series Single Board Computer) Installation and Use*” , 2001