□ 기술해설 □

재구성 가능한 객체 기반 실시간 운영 체제 커널

삼성종합기술원 유광현* · 송영근** · 최익수** · 이경훈** · 김경희**
김호일** · 박정형** · 김동찬* · 채종원**

● 목 차 ●

1. 서 론

실시간 응용의 경우 간단한 가전 제품과 같이 소수의 제어점을 갖는 응용으로 부터 공장 자동화를 위한 복잡한 응용에 이르기까지 다양한 응용 분야가 존재한다. 따라서, 실시간 운영 체제 커널의 경우 다양한 응용을 모두 만족시키는 운영 체제의 자원(resources)관리 정책을 개발하는 것은 매우 힘든 일이다. 예를 들면, 실시간 스케쥴링 정책의 경우 FIFO(First In First Out)[2], RM(Rate Monotonic)[3], EDF(Earliest Deadline First)[4]를 비롯하여 많은 알고리즘이 개발되었으나, 일상적인 가전 제품의 제어에서 공장 자동화 기기까지 모두 사용할 수 있는 알고리즘은 알려져 있지 않다.

따라서, 실시간 응용의 다양한 요구에 맞는 서비스를 효과적으로 제공하는 방법중의 하나는 실시간 운영 체제 수준에서 응용의 요구에 따라 자신이 제공하는 서비스를 적응(adapt)시킬 수있는 적응력(adaptability)이 필수 요건이라고 할 수 있다. 실시간 응용 수준에 적응성을 부여키 위해서는 실시간 커널내의 자원에 대한 관리 정책을 사용자 수준에서 제어할 수 있는 기법을 제공하여야 한다. 자원 관리 정책은 커널이 관리하는 처리기, 메모리, 프로세스간 통신 등의 서비스를 유지, 관리하는 기법으로서, 처리기 스케쥴링 기법에서 볼 수 있듯이 응용 프로그램의 수행 속도에 많은 영향을 미친다. 따라서, 응용 프로그램이 커널의 자원 관리 정책에 개입하여 자신의 자원 사용 패턴에 가장 적합한 관리 정책을 선택할 수 있다면 최적화된 동작 환경을 형성할 수 있을 것이다. 커널에 대한 재구성은 커널내 자원 관리 정책을 구현한 소프트웨어 모듈의 재구성으로 구현될 수 있다. 재구성은 소프트웨어 모듈에 대한 추가, 대치, 삭제 등의 연산[1]을 의미하는 것으로, 커널내 모듈에 대한 재구성은 자원 관리 정책에 대한 변경을 의미하게 된다.

본 연구진이 개발중인 시스템은 객체를 기반으로 한 실시간 커널, 커널 및 응용 수준의 객체를 대상으로 사용자가 구성(configure)할 수 있도록 지원하는 구성 도구, 그리고 실시간 응용의 개발을 지원하는 개발 환경으로 구성되며, RISC계열의 처리기를 대상으로 그 프로토타입 시스템을 개발 중에 있다.

본 논문의 2장에서는 운영 체제의 설계 부분

*비 회 원
**정 회 원

을, 3장에서는 프로토타입 시스템 전체 구조, 4장에서는 커널내의 각 객체의 기능 설계, 5장에서는 구성 도구 및 개발 시스템에 관한 내용을 기술하고, 6장에서는 논의점을, 마지막으로 7장에서는 결론을 기술한다.

2. 재구성 가능한 실시간 운영 체제의 설계

운영 체제 커널 및 개발 환경을 설계하면서 다음과 같은 점에 목표를 두었다.

- 다양한 응용에 대한 커널의 적응성
- 탄력성(flexibility)
- 다양한 자원 관리 정책의 제공
- 일관성(uniformity)
- 확장성(expandability)

따라서, 본 연구진은 다음과 같은 특징을 갖는 시스템을 설계하였으며, 그 프로토타입 시스템을 구현 중에 있다.

- 객체 기반 시스템
  실시간 응용을 구성하는 커널 및 응용 소프트웨어들을 모두 객체로 모델링하는 기법을 사용하였다. 이때 모든 객체는 재구성의 단위체를 형성하며 구성 도구의 지원을 받아 최적화된 최종 목표 시스템을 구성하게 된다.
- 다양한 종류의 실시간 스케쥴링 기법의 지원
  실시간 스케쥴링 기법을 구현한 다수의 스케쥴러를 제공하여 응용 수준에서 자신에 가장 적합한 스케쥴러를 선택할 수 있도록 지원한다. 또는 자신이 작성한 스케쥴러를 대신 직접 사용할 수도 있다.
- 탄력적인 메모리 관리 시스템을 지원
  실시간 응용에서 필요한 메모리 관리 정책을 지원함과 아울러 복잡한 응용과 디버깅의 지원을 위하여 가상 주소 공간 서비스를 제공한다.
- 실시간 요구 사항의 만족
  상기한 기능을 지원하면서도 실시간 요구 사항들—예를 들면 최악 경우 인터럽트 지연 시간(worst-case interrupt latency time), 시스템 호출 부담(system call overhead), 인터럽트 복귀 시간(interrupt return time), 문맥 교환 시간(context switch time) 등—을 만족하도록 한다. 또한, 재구성 가능한 실시간 스케쥴러와 개발 환경의 시간 분석 도구를 이용하여 응용의 실시간 요구 사항의 분석을 위한 기반을 제공한다.

3. 전체 구조

본 연구진에서 설계한 실시간 운영체제의 구조는 다음의 그림 1에 나타나 있다. 커널은 시스템에서 제공하는 자원을 효율적으로 재구성할 수 있도록 객체에 기반한 설계 기법을 사용하여 설계하였으며, 객체를 기초로 하여 재구성 작업을 용이하게 수행할 수 있는 구조를 갖는다.

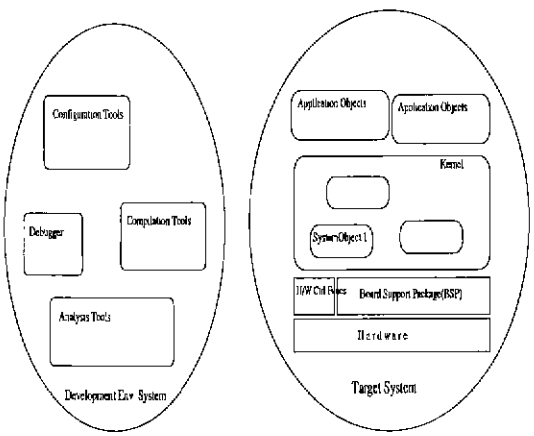

그림 1 제안된 시스템의 전체 구조

개발환경은 크게 나누어 실시간 소프트웨어의 개발을 지원하는 도구들과 실시간 소프트웨어의 요구 사항에 따라 운영 체제의 기능을 최적화하는 구성 도구로 나뉜다. 개발 도구는 응용 프로그램의 개발을 위한 컴파일러, 디버거 등의 기본 개발 도구이외에 사건 분석기(event analyzer), 시간 분석 및 표시기(time analyzer & display), 시뮬레이터(Simulator) 등이 있다. 구성도구는 본 실시간 운영 체제 과제의 핵심이 되는 부분의 하나로서, 작성된 응용의

특징에 따라 커널 혹은 기타 소프트웨어 모듈을 분석하여 이를 응용에 최적화된 형태로 구성할 수 있도록 지원하는 도구이다.

커널은 실시간 커널에서 제공하는 기능을 객체로 구성하여 각 기능이 재구성의 단위가 될 수 있도록 설계하였으며, 커널에서 제공하는 기능을 크게 나누면 다음과 같다.

- 실시간 계산을 위한 계산 활동의 지원
- 효과적인 메모리 관리를 지원
- 간단하면서도 효과적인 동기화 기능 및 통신 기능 제공
- 커널내의 서비스에 대한 객체화된 인터페이스 및 서비스 제공

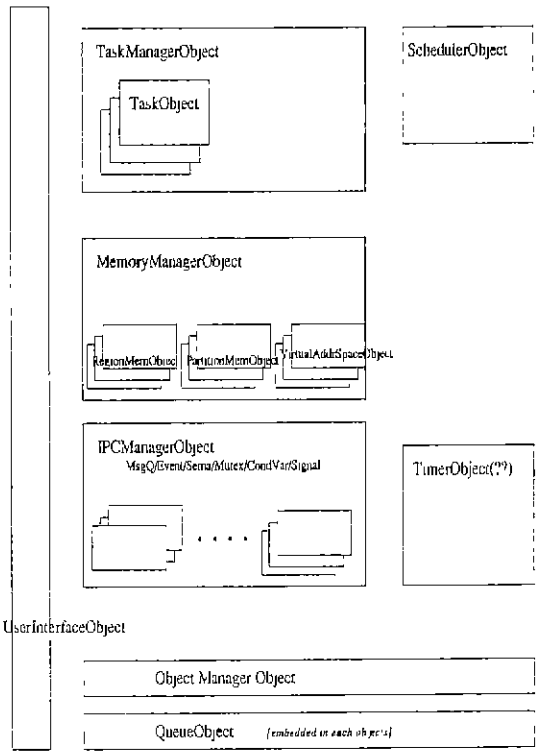

그림 2 커널내 객체들

그림 2는 본 연구진이 설계한 커널내의 객체와 그 관련성을 보이고 있다. 객체는 객체에 대한 관리자 객체와 서비스에 대한 정보를 유지하는 서비스 객체로 나뉜다. 객체 관리자 객체는 객체에 대한 지명(naming) 서비스를 제공하며 객체에 대한 관리 정보를 유지 관리한다. 태스크 관리자 객체(TaskManagerObject)는 태스크 객체(TaskObject)에 대한 생성 및 제거 기능을 담당하며, 스케쥴러 객체와 협동하여 스케쥴링 기능을 제공한다. 스케쥴러 객체(SchedulerObject)는 태스크 관리자 객체로부터 태스크 객체를 전달받아 자신이 관리하는 객체에 대한 스케쥴링 기능을 담당한다. 또한, 각 객체의 수행 상태에 따라 스케쥴링 큐(queue)들을 관리하고, 타이머 인터럽트를 받아 스케쥴 연산을 수행한다. 메모리 객체는 다양한 크기의 메모리 조각(slice) 형태로 할당/반환할 수 있는 리전 객체(RegionObject)와 일정크기의 메모리 조각 형태로 사용할 수 있는 파티션 객체(PartitionObject), 그리고 가상 주소 공간을 나타내는 가상 주소 공간 객체(VirtualAddrSpaceObject)로 구성된다. 그리고, 타이머 시간 및 실시간 클럭(Real Time Clock)을 관리하는 타이머 객체(TimerObject)가 있다. 사용자 시스템 호출시, 인터럽트의 발생시 이를 해석하고 처리하는 사용자 인터페이스 객체(UserInterfaceObject)가 있다.

그림 2에 보인 객체들은 외부 인터페이스가 같은 다른 객체에 의하여 언제든지 대체될 수 있으며, 본 연구진에서는 재구성 기능을 지원키 위한 개발 환경의 구성도구 또한 개발 중에 있다.

4. 커널내 객체

위에서 살펴본 바와 같이 samRTOS는 객체지향형 운영체제를 목표로 하고 있다. 이러한 객체지향 운영체제는 그 운영체제의 각 기능에 관련된 객체를 선언하고 그 객체에 따른 정적/동적 특성을 정의하며 객체간 관계를 규명함으로써 기술될 수 있다. 따라서, 본 절에서는 samRTOS의 객체들의 종류와 기능을 OMT[5, 6, 7]방법으로 설명하고자 한다.

4.1 Task관련

Task관련 객체들은 시스템을 적절히 운영하기 위한 각 task들을 제한 시간내에 수행 되도록 하기 위한 그들 간의 상호 작용을 제어하는 스케쥴러와 task 자체의 생성/삭제 등을 관장하는 모듈들에 관하여 정의되어 있다. 이런

task관리는 상태의 전이, 각종 큐의 관리, task들의 실행, 생성, 삭제, 지연 등의 기능을 수행하고 있다. 이러한 task관리부를 설계하기 위해 윗 절에서 언급한 설계 정책에 맞추기 위해 재구성성, 유연성, 확장성, 모듈화 등에 촛점을 맞추어 객체를 설계하려고 했으며, 마이크로 커널 구조를 고려하였다.

4.1.1 내부 구조

samRTOS는 자원의 관점에서 ready queue와 TCB, task address space, entry point가 있다. 이를 간략히 살펴보면 상태전이는 상호작용하고 있는 task들이 특정 시점에서 어떤 task가 무슨 상태에 있는 지를 나타내도록 스케쥴러에 의해 결정된다. 대개 이것은 실행 중인 상태에서 다른 상태로 변화되었을 때 실행이 중단된 시점의 내용을 가지고 있다가 실행이 계속될 때 복구된다. 이러한 전이는 스케쥴러의 행동을 표현하며 다른 모듈과의 연관성을 결정한다. samRTOS는 실행 중인 task를 나타내는 running, Timer에 의한 delayed, 실행하기 위해 기다리는 ready 그리고, 세마포어, 메세지, 이벤트 등과 같은 것에 의해 기다리는 blocked상태가 있다. TCB는 task를 실행하기 위한 필요한 정보를 가지고 있는데 커널 내부에서 관리되며 운영체제의 설계 정책을 나타내고 있으며, samRTOS에서는 재구성성으로 인해 스케쥴링 정책에 따라 TCB 내용이 바뀐다.

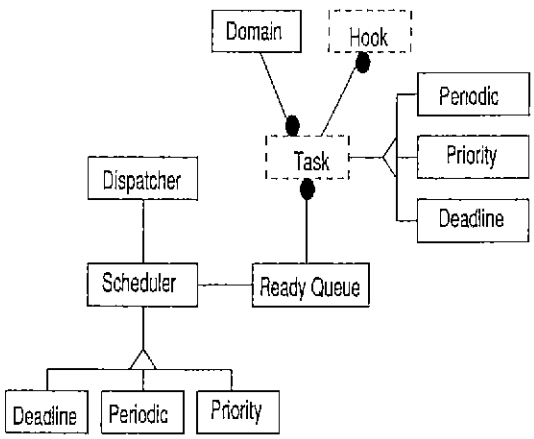

그림 3 Task의 정적 모델

스케쥴러 진입점은 각 서비스가 끝난 후 스케쥴러로 제어가 오기 전에 수행을 하기 위한 부분으로 하드웨어와 밀접한 시스템 자원의 관리와 같은 스케쥴링의 선/후처리를 담당한다. 이는 독립적인 객체로는 존재치 않고 서브루틴으로 구성되어 있다. 한편, task address space는 memory 객체에서 기술하려고 한다.

4.1.2 객체 모델

task관리부를 구성하는 요소들을 나열하고 그 요소들 간의 관계와 자료구조를 나타내는 task 모델을 보여주고자 한다.

4.2 Memory관련

실시간 운영체제의 multi-tasking환경에서 메모리 관리부는 시스템이 갖는 메모리 자원을 관리하여 커널과 각 task가 수행되기 위해 필요한 작업 공간을 적절한 시간내에 할당하고 잘못된 사용으로부터 메모리를 보호하며 사용이 끝난 메모리 공간을 회수하여 재사용될 수 있도록 한다. 또한, task들간에 공유될 수 있는 영역을 제공한다. 그런데 일반적인 것과 달리 실시간 운영체제의 메모리 관리부는 메모리 자원의 요청에 대해 실시간으로 처리 하는데 중점을 두며 필요한 경우 가상 메모리를 지원할 수 있는 구조로 설계하려고 했다. 따라서, 예측가능한 실시간 서비스와 가상 메모리의 편리한 확장성 그리고 실메모리이든 가상메모리이든지 여부에 관계 없이 동일한 인터페이스를 지원하려고 한다.

4.2.1 내부 구조

이러한 메모리 관리부는 메모리 블록을 제공하는 실메모리 관리층이고, 가상메모리를 지원하는 가상메모리 관리층이며, 2가지 메모리 시스템을 동일한 인터페이스로 사용할 수 있도록 지원하는 인터페이스층이다. 실메모리만을 지원할 때는 2개로, 가상메모리를 지원할 때는 3개 모두로 구성된다.

samRTOS에서는 이 구조를 지원하기 위해 메모리를 정적인 부분과 동적인 부분으로 나누었다. 정적인 부분은 다시 ROM화 가능부와 불가능부로 나눌 수 있고, 동적인 부분은 시스

템 시작 시점에 생성되는 부분과 실행중에 생성되는 부분으로 나눌 수 있다. 모든 task는 자신의 메모리 공간인 도메인을 갖는다. 이 도메인은 하드웨어 MMU를 사용하는 것과 같이 논리 주소를 사용하는 경우 task의 독립적인 주소 공간이며, 논리 주소를 사용하지 않을 때는 해당 task가 사용하고 있는 주소 공간이라고 할 수 있다. 이 도메인은 해당 task가 사용하는 모든 리전들로 구성되며 하나의 task는 여러 개의 리전을 할당 받아 가질 수 있다. 또한, 리전은 서로 다른 task들 사이에서 공유되고 보호되는 기본 대상이다.

4.2.2 객체 모델

위에서 설명한 samRTOS의 메모리 관리부를 객체로 기술한 것이 객체들의 관계를 그린 객체 모델들이다. 정적 모델에서는 구성 요소들에 대해 클래스간의 관계도로 나타내고 해당 클래스들의 자료 구조와 기능을 클래스 정의 관점에서 기술한다. 다음 그림은 메모리 관리부의 memory 모델이다.

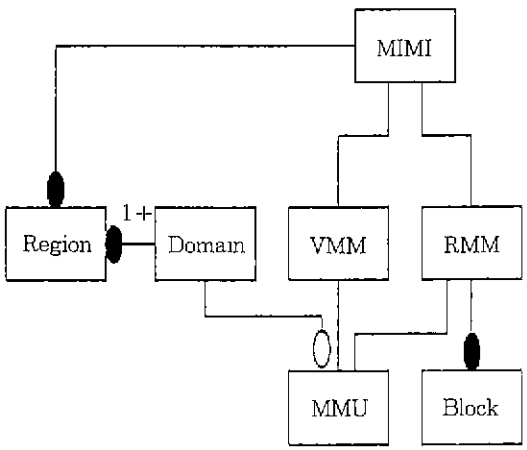

그림 4 메모리의 정적 모델

4.3 IPC관련

multi-tasking환경에서 2개 이상의 태스크간 동기화와 데이터 전송 및 통신을 위해 많은 IPC 기법들이 사용된다. IPC 기법들은 그 특성에 따라 동기화를 위한 것과 데이터 교환을 위한 것으로 나뉜다. 실시간 운영체제에서 사용되는 IPC 기법들은 기존의 IPC 기법들에 대해 시간적인 제약성 및 예측 가능성을 제공할 수 있어야 한다. 또한, 다양한 응용 분야에 따라 적절한 IPC 기법을 제공할 수 있도록 설계하였다. 따라서, 기존에 사용되고 있는 IPC 기법들을 모두 지원할 수 있는 대규모의 IPC 기법군을 설정하고, 이들 중 필요에 따라 일부 기법들을 추가/삭제가 가능하도록 구조를 설계하였다. 이를 위해 다양성, 사용자 정의성, 확장 가능성, 시간 제약성 그리고 예측 가능성을 갖는 IPC 기법을 제공한다.

4.3.1 내부 구조

설계된 IPC 관리부는 다양한 IPC 기법들을 이루는 여러 클래스간의 상관 관계와 운영체제 내의 다른 모듈들과의 연관성을 나타낼 수 있도록 구성된 IPC 모델로 구성된다. 이 모델은 IPC 기법에 대한 개념적 단계를 나타내는 IPC 클래스를 중심으로 구성된다. IPC 클래스는 사용자 수준의 태스크와 연관될 뿐만 아니라 Timer 클래스, Trap 클래스, Waiting Queue 클래스 등을 포함하는 커널내의 다른 모듈들과도 관련된다. 이의 관계는 아래 그림의 상단에 나타나 있다. IPC 클래스는 기능에 따라 Communication 클래스와 Synchronization 클래스로 구분된다.

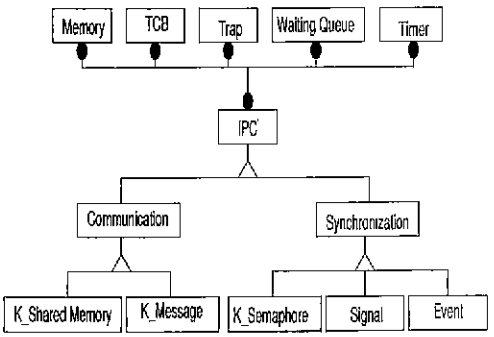

그림 5 IPC의 정적 모델

4.4 Timer관련

시스템 시간은 운영체제의 모든 동작을 동기화하고 스케쥴의 단위를 결정하는 이외에도 그 자체가 메모리와 같은 자원이라는 관점에서 다양한 서비스를 제공한다. 특히 시스템 동작의

동기화와 동작의 유효성을 결정하는 척도가 된다. 이 Timer 서비스는 몇가지 설계 개념을 갖는다.

클럭 인터럽트는 대개 상위 인터럽트 순위를 차지하며 발생 빈도가 잦은 것이기 때문에 이를 서비스하기 위한 시간을 최소화 하는 것이 전체 시스템 성능에 영향을 덜 주므로 서비스 시간을 최소화한다. 다음은 Timer의 확장을 위해 3가지 서로 다른 속성을 갖는 클래스로 구분하고 필요한 기능의 특성에 따라 상속 관계를 정립하여 확장성을 제공하였다. 예를 들면, IPC의 기법중 Timeout특성이 제외된 운영체제에서는 타이머 기능에 영향을 주지 않고 IPC Timer만을 제외 시킬 수 있다.

4.4.1 내부 구조

시간 관리부에서 제공하는 주요 기능은 시스템 시간을 유지하여 전체 시스템에 제공하고, 시스템의 정상 동작 여부를 검사하며, 클럭 인터럽트를 처리하고 스케쥴링과 관련된 task의 시간을 조절하며 커널의 다른 콤포넌트나 응용 task에서 요구하는 타이머기능이 있다. 시간 관리부에서 시간은 캘린더 시간과 클럭 시간이 있다. 캘린더 시간은 시스템의 시각을 유지하는 것으로 하드웨어에 의해 제공되는 서비스이고 클럭 시간은 시스템이 초기화된 후에 그 시점까지의 경과시간 으로 일정한 주기로 반복되는 하드웨어 클럭 인터럽트를 처리하여 스케쥴링 및 다른 콤포넌트의 시간 자원으로 사용한다. 한편 시간 관련 서비스 요청은 절대/상대적인 시간으로 이뤄지는데, 절대 시간은 현재의 시각으로 지금까지의 경과 시간으로 해당 자료구조의 값이며, 상대 시간은 특정 시점으로부터 시간 간격을 말한다.

4.4.2 객체 모델

위와 같은 구조를 지원하기 위해 하드웨어에서 제공하는 RTC를 관리하는 RTC 클래스와 시스템 동작의 이상 유무를 확인하는 WatchdogTimer 클래스, 하드웨어 클럭 인터럽트를 처리하는 ClockTick 클래스, 응용 Task나 시스템 task에서 사용하는 timer를 제동하는 Timer 클래스가 제공된다. 이러한 구성을 아래에 그렸다.

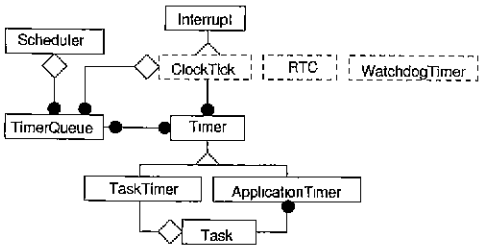

그림 6 Timer의 정적 모델

4.5 Exception관련

예외 관리부에서는 정상적인 운영체제의 제어에서 벗어나 인터럽트나 예외를 처리하고 사용자 task로부터의 시스템 호출을 수행하기 위한 트랩을 처리한다. 따라서, 하드웨어와 밀접한 관계에 있으면서 운영체제 전체 성능에 영향을 미친다. 그래서, 이 관리부를 설계함에 있어서 하드웨어 관련부의 추상화와 선점 지연 시간의 최소화, 사용자에게 편리한 핸들러 작성법 제공, 운영체제와 사용자 환경과의 독립성을 고려 하였다.

4.5.1 객체 모델

예외, 트랩, 인터럽트는 운영체제의 정상적인 제어를 벗어나 하드웨어의 영향을 받는 예외 분기에 해당한다. 이러한 이유로 그들 각각을 ExceptBranch의 구체화된 형태인 상속 관계로 정의하였다. ExceptBranchDispatcher 클래스는 모든 예외 분기를 처리하는 데 공통적인 부분에 관계 되어 있다. Kernel 클래스는 운영체제의 초기화시 혹은 기타 필요한 정보를 추

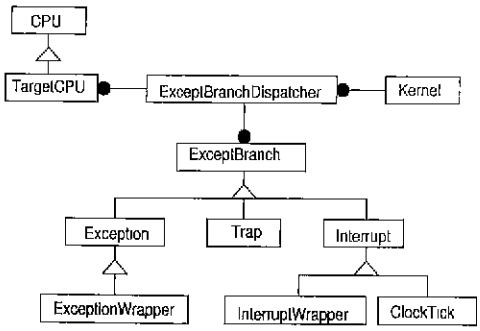

그림 7 Interrupt의 정적 모델

상화 한 것이며, CPU 클래스는 일반적인 프로세서를 추상화 한 것이다. 사용자가 쉽게 핸들러를 작성할 수 있도록 InterruptWrapper 클래스와 ExceptionWrapper 클래스가 있다.

5. 개발 환경

OS Configuration Tool의 목적은 다양한 OS needs를 해결하기 위한 방안으로 OS의 특성들을 사용자가 그 응용분야에 적합하게 선택할 수 있도록 OS의 재구성 방법을 제공하는 것이다. 즉, RTOS를 개발하는 platform을 제공하고, domain-specific한 OS 특성 조합의 guildeline을 제공함으로써 결과물인 실시간 운영체제의 성능을 향상시키는 것이 목적이다.

이를 위한 OS구성의 단계는 1)사용자가 선택한 OS 특성들 조합의 가능성 여부와 유효성을 검증하는 단계, 2)사용자의 요구를 정형화된 DSL로 받아들여 명세자체를 분석, 검증하는 단계, 3)사용자의 요구를 분석하여 이를 바탕으로 RTOS kernel의 동작을 simulation하는 단계로 개발될 수 있다.

구성의 대상이 되는 것은 사용자가 사용 가능한 system call의 종류를 선택하는 것으로부터 kernel의 각 component, OS의 server, system parameter, hardware dependent한 부분의 선택, kernel의 정책(policy)을 지원하기 위한 구현 mechanism 수준까지 다양하게 존재한다. 뿐만아니라 kernel의 특성이나 응용분야에 따른 OS구성의 guideline을 포함한다.

OS configuration tool에서 검증의 대상으로 삼는 것은 결과물인 OS의 논리적/물리적 동작의 유효성 여부와 실시간성의 보장, 즉 사용자가 요구한 시간 제약조건(time constraint)의 만족 여부 문제이다.

이러한 OS configuration tool을 위해 객체의 개념을 OS kernel에 도입하고, 각 재구성의 대상이되는 component들의 관계를 규정하는 rule을 정의한다. 구성의 대상 component들은 Module library에 저장되어 library manager에 의해 관리되며 Coordinator의 분석된 정보에 따라 사용자가 요구한 명세에 해당하는 특성을 지닌 RTOS로 생성된다. Coordinator는 사용자 선택에 따라 선택된 OS 구성요소들의 제어/자료의 흐름이 유효한가의 여부를 결정하며 이를 위해 component들 사이의 논리적/물리적 관계를 유지/관리한다. component들 사이의 관계를 기술하는 rule에 따라 적절히 분류된 component들의 정보가 계층화된 구조로 저장된다. OS Configuration Tool의 전체 구조는 아래 그림과 같다.

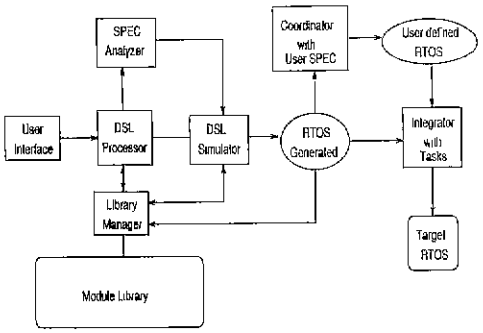

그림 8 Configuration Tool의 구조

6. Discussion

본 논문에서 사용한 객체를 기반으로 커널을 구성하는 기법은 기존 커널에 비하여 다음과 같은 장점이 있다.

- 커널의 자원 관리 정책이 자연스럽게 모델링 될 수 있다.
- 커널의 서비스가 객체 형태로 모델링되므로, 구성(configuration)이 용이하고, 재구성시 주변 효과가 적어 쉽게 서비스를 재구성할 수 있다.
- 새로운 기능의 확장시 새로운 기능을 갖는 객체를 추가만 하면 되므로 시스템 확장이 쉽다.
- 객체로 모델링된 커널일지라도 객체지향형 언어로 구현하지 않아 그러한 언어에서 발생하는 비결정성을 없앰으로써 실시간성을 보장한다.
- 다양한 용도에 맞도록 커널을 재구성할 수 있으므로 실시간 응용에 맞는 여러 실시간 스케쥴러 등과 같은 실시간 보장 알고리즘을 채택하여 실시간성을 만족 시킨다.

7. 결 론

본 논문에서는 현재 삼성종합기술원 RTOS 팀에서 개발중인 실시간 운영 체제의 구조에 대하여 기술하였다. 개발중인 운영 체제는 객체를 기반으로 하여, 사용자의 요구에 따라 재구성 가능한 운영 체제 커널과 구성 및 개발을 지원하는 개발 환경으로 구성된다.

객체를 기반으로 커널을 구축한 기법은 커널의 기능을 재구성함에 있어서 매우 용이하고, 구조적인 기법을 제공할 수 있다. 또한, 시스템 기능에 대한 확장성이 우수하고, 상황에 따른 적응성이 뛰어난 커널을 생성할 수 있는 장점이 있다.

참고문헌

[1] C. Hofmeister and J. Purtilo. Dynamic Reconfiguration in Distributed Systems : Adapting Software Modules for Replacement. In *Proceeding of 13th International Conference on Distributed Computing Systems*, pages 101-110, May 1993.

[2] Nain, P.;Towsley, D. Comparison of hybrid minimum laxity/first-in-first-out scheduling policies for real-time multiprocessors In *IEEE Transactions on Computers*. vol.41, no. 10., p. 1271-8., Oct. 1992., CODEN : ITCOB4., ISSN : 0018-9340.

[3] M.H.Klein, T.Ralya,B.Pollak. *A Practutiner's Handbook for Real-Time Analysis : Guide to Rate Monotonic Analysis for Real-Time Systems*. Kluwer Academic Publishers,Boston, MA, 1993.

[4] Ferrari, A.D. Real-time scheduling algorithms *Dr. Dobb's Journal*., vol.19, no.15., p. 60, 62, 64, 66, 94, 96., Dec. 1994., CODEN : DDJSDM., ISSN : 1044-789X.

[5] Schneeweiss, H. Shlaer/Mellor or Rumbaugh? A discussion of two popular object-oriented methods. *Ada in Europe. First International Eurospace-Ada-Europe Symposium Proceedings., p. 155-61.*, 1994.

[6] Wooding, T.;Plant, N. Choosing a methodology (for OO systems analysis and design). *Object Manager.*, p. 6-8., Oct. 1994.

[7] Thuraisingham, B.;Schafer, A. RT-OMT : a real-time object modeling technique for designing real-time database applications. *Proceedings of the IEEE Workshop on Real-Time Applications (Cat.No.94TH0663-5).*, p. 124-9., 1994.

유 광 현

1996 서울대학교 컴퓨터공학과 박사
1996~현재 삼성종합기술원 S/W연구실 선임연구원
관심분야 : 운영 체제, 분산 처리 시스템, 실시간 운영 체제

송 영 근

1993 중앙대학교 전자계산학과 박사
1984~90 삼성전자 정보통신 연구소 연구원
1995~현재 삼성종합기술원 S/W연구실 수석연구원
관심분야 : Distributed system, Real time operating system, Software Configuration and Formal/Category theory

최 익 수

1990 서울대학교 컴퓨터공학과 석사
1990~현재 삼성종합기술원 S/W연구실 선임연구원
관심분야 : 분산 처리 시스템, 실시간 운영 체제, 객체지향 모델링

이 경 훈

1993.3~95.2 고려대학교 전산과학과 석사
1995.2~현재 삼성종합기술원 S/W연구실 주임연구원
관심분야 : 실시간 운영체제, Mobile Computing, 멀티미디어

김 경 희

1995 포항공과대학교 전자계산학과 학사
1995~현재 삼성종합기술원 S/W연구실 연구원
관심분야 : Real time operating system

김 호 일

1984.3~90.2 전남대학교 전산통계학과 학사
1990.12~현재 삼성종합기술원 S/W연구실 전임연구원
관심분야 : real-time systems, object-oriented software engineering

박 정 형

1993 연세대학교 전산과학과 석사
1993~현재 삼성종합기술원 S/W연구실 전임연구원
관심분야 : 운영 체제, 실시간 운영 체제, 소프트웨어 모델링 및 분석

김 동 찬

1986 한국항공대학교 전자공학과 학사
1989~94 삼보컴퓨터 연구원
1994~96 삼성종합기술원 S/W연구실 전임연구원
관심분야 : 실시간 운영 체제

채 종 원

1984.2 한국항공대학교 전자공학과 석사
1983.10~86.10 삼성전자 통신연구소
1986.11~현재 삼성종합기술원 S/W연구실 수석연구원
관심분야 : Distributed System, Real Time System

KOREA-JAPAN Joint Workshop on Algorithm and Computation

- 일 자 : 1996년 8월 23~24일
- 장 소 : 한국과학기술원 전산학과
- 관련분야 : Automata, Languages, Computability
  Combinatorial/Graph/Geometric/Randomized Algorithms
  VLSI/Parallel Algorithms, Networks/Distributed Algorithms
  Theory of Learning/Robotics, Number Theory/Cryptography
  Graph Drawing, Computational Logic
- 주 최 : 한국정보과학회 컴퓨터이론 연구회
  위원장 장직현 교수(서강대 전산학과)
  jchang@alglab.sogang.ac.kr
- 문 의 : 신찬수(KAIST 전산학과)
  cssin@jupiter.kaist.ac.kr
  Tel : 042-869-3553 Fax : 042-869-3510