J · BOSN RTOS 개발하기

마이크로 커널(TIME SERVER)

J · BOSN 실시간 운영체제는 고성능의 멀티-태스킹(Multi-tasking)을 지원하고 매우 강력한 실시간 응용프로그램을 개발할 수 있는 바탕을 제공하여 준다. 고성능의 실시간 커널(나노커널)과 다른 서비스모듈(마이크로커널, 매크로커널)이 결합되어 전체시스템이 이루어진다. 모든 모듈은 각자의 독립성을 최대한 보장하고 있으며, 각 모듈은 실시간 시스템의 서비스를 확장시킨다. JBOSN 운영체제에서 기본적으로 제공하는 모듈 등을 통해 실시간 운영 체제에 대해 자세히 소개하고자 한다. 지난호 나노커널에 이어, 이번호는 이러한 특성을 구현하기 위한 근본적인 개념을 담고 있는 시간 서버의 설계와 구성 원리를 살펴보겠다.

연재 차례

JBOSN RTOS의 구조는 **그림 1**에 나타나 있다, 구조적인 측면에서 보면 그림 1에서 나타나듯이 계층적이고 모듈화되어 있는 각 모듈 중에 JBOSN RTOS의 시간과 관련된 기능을 제공하는 목적으로 제작된 것이 시간서버(Time Server)이다. 운영체제의 측면에서 보면, 시간서버는 운영체제의 시간관리 및 전체적인 시간동기화를 목적으로 하여, 전체 JBOSN RTOS의 멀티태스킹을 구현하고, 시스템전체의 시간을 관리하는 역할을 담당하여 운영체제에 필수적인 부분이다.

이번호는 이러한 특성을 구현하기 위한 근본적인 개념을 담고 있는 시간 서버의 설계와 구성 원리를 살펴보겠다.

시간 서버(Time Server)

시간서버는 시간을 매우 중요한 자원으로 관리하는 실시간 운영체제에서 필수적인 기능과 JBOSN RTOS만의 독특한 기능을 제공하여 주는 JBOSN RTOS의 마이크로커널 서버중 하나이다. 나노커널에서 제공하는 시간과 관련된 기능은 Tick Timer와 관련된 인터럽트를 시간 서버에 전달하는 것이다. 하

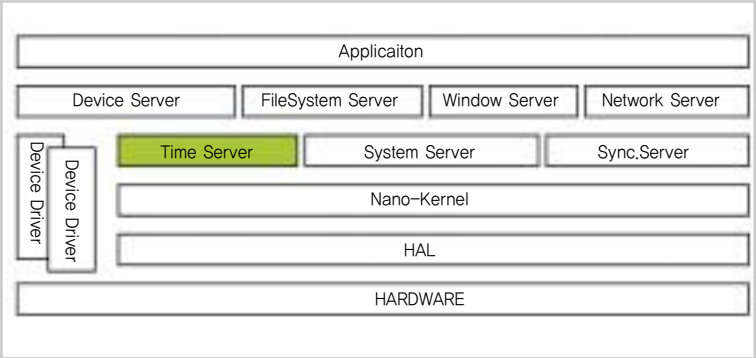

그림 1. JBOSN RTOS 구조

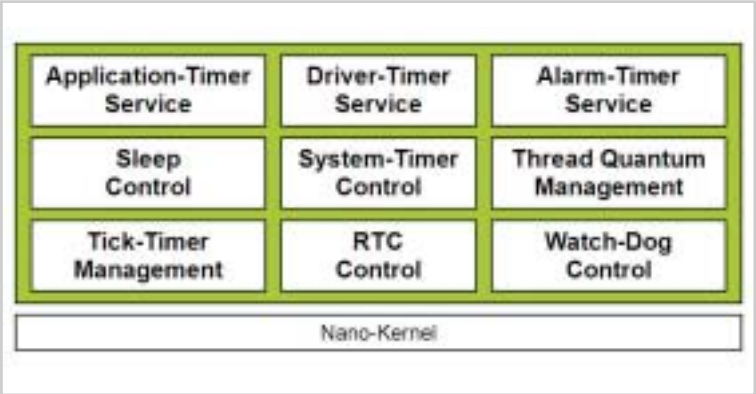

그림 2. 시간서버 구성도

드웨어와의 연결은 나노커널에서 제공하는 라이브러리를 호출하여 제어한다. 물론 HAL 레이어에서 하드웨어를 제어할 수 있는 함수를 제공하여 준다. 시간서버는 그림 2처럼 구성이 되어 있다. 하단의 나노커널과 직접적으로 연결되어 기능을 수행하는 부분은 Tick-Timer management, RTC-control, Watch-Dog Control의 세부분으로 나누어져 있다. 그 외의 다른 부분은 모두 Tick-Timer Management 부분의 현재 Tick 값(MsTickCount)을 사용하여 구현이 된다. 즉, 시간 서버의 최대 분해능(resolution)은 Tick Timer 값으로 한정이 된다.

예를 들면, Tick Timer 값을 1/1000초(1ms)로 한다면, 시간서버의 다른 모든 기능은 1ms이하의 시간은 측정할 수 없고, 1ms의 배수의 값으로만 시간 서비스를 제공할 수 있다. 시스템-타이머, 쓰레드 퀀텀 관리는 운영체제 내부동작에 많이 관련된 함수이다. 외부의 사용자는 사용하지 않는 부분이다. 그러나 애플리케이션 타이머 컨트롤, 드라이버 타이머 컨트롤, 알람-타이머 컨트롤과 슬립 컨트롤(Sleep Control) 부분은 디바이스 드라이버, 응용프로그램과 다른 운영체제를 이루고 있는 서버 및 사용자가 만든 서버에서 사용이 된다. 다음은 각각의 부분에 대하여 자세히 살펴보겠다.

왓치-독 컨트롤

하드웨어의 왓치-독(Watch-Dog)을 제어하여 서비스를 제공하는 부분이다. 왓치-독은 매우 특별한 장치로서 시스템의 정상적인 동작 상태를 확인하여, 시스템을 재시작할 것인지를 판별하는 부분이다. 시스템 설계자가 매우 조심스럽게 다루는 부분으로 시스템의 정상동작을 판별하는 것이 매우 중요하다. JBOSN RTOS는 시스템의 정상동작 상태를 획일적으로 측정하지 않고, 사용자의 몫으로 남겨두었다. 사용자가 이 부분을 사용할 수도 있지만, "KernelIo Control" 함수를 호출하여 임의적으로 왓치-독 타이머를 제어할 수 있는 방법을 채택하여도 좋다.

RTC Control

하드웨어의 RTC(real-time controller)는 날짜를 매우 정확하게 제공하여주는 장치이다. JBOSN RTOS는 현재의 날짜를 응용프로그램에 전달하거나 응용프로그램에서 현재 날짜를 설정할 수 있도록 해 준다. 날짜는 JBOSN RTOS의 SYSTEMTIME이라는 구조체로 표현이 되며, 1/1000초의 기

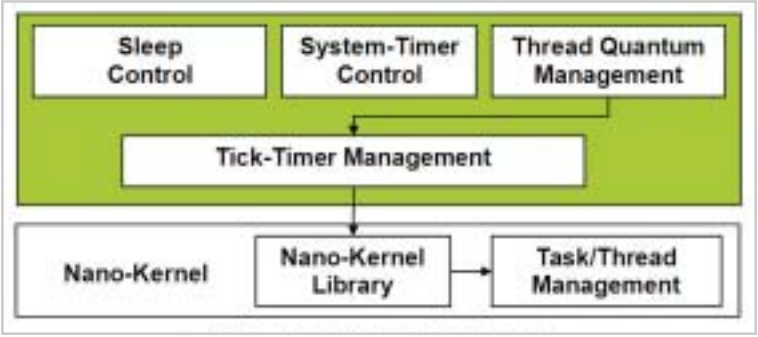

그림 3. Thread Quantum Management 호출도

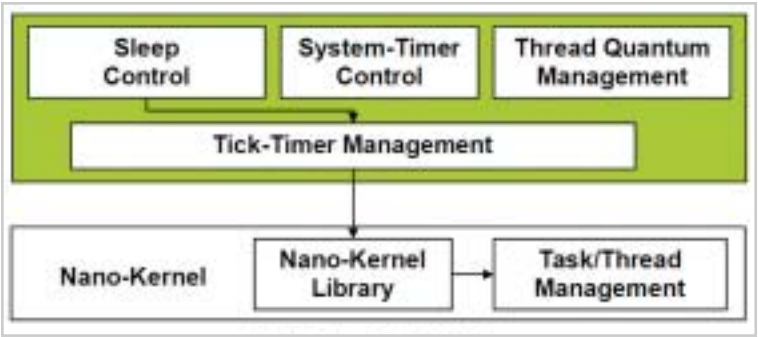

그림 4. Sleep Control 호출도

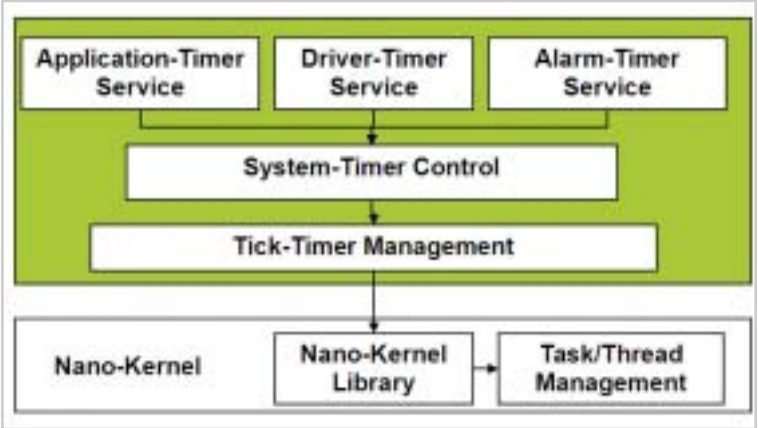

그림 5. Sleep Control 호출도

본단위를 가지고 있다. SYSTEMTIME이라는 구조체는 현재 사용하는 년/월/일 형태의 구조가 아니어서 보기가 어렵다. HUMANTIME이라는 구조체를 제공하여 년/월/일 형태의 구조체를 제공하고 있으며, 위 두개의 구조체를 서로 변환할 수 잇는 라이브러리를 제공하여 준다. 실제하드웨어와 연결하는 부분은 HAL부분에서 반드시 제공이 되어야 한다 (Porting Guide참조).

Tick-Timer Management

HAL 부분에서 제공하는 Tick-Timer 인터럽트를 나노커널에서 받아들여 시간서버에게 관련된 정보를 전달하여 준다. 이 정보를 직접적으로 전달받아서 Tick-Timer Managemen 부분이 동작하게 된다. 즉, 나노커널에서 Tick-Timer 관련부분이 새로 갱신이 될 때마다 이 부분이 실행된다. 시간서버의 나머지 부분이 여기서 나오는 시간정보를 이용하여 시간과 관련된 모든 서비스의 동기를 맞추게 되므로 JBOSN RTOS의 핵심부분이다.

Tick-Timer management부분은 매 Tick마다 수행이 되면 시스템에 상당한 부담이 된다. 따라서 필요할 때만 수행이 되도록 설계를 하면, 가장 이상적으로 시스템의 부하를 줄일 수 있다. 나노커널에서 Tick-Timer management 부분에 필요할 때 만 시간정보를 전달하기 위해서는 시간 서버에서 미리 나노커널에 필요한 시기(Tick Timer 전달 시기)를 설정하여 놓는다. 예를 들면, 현재 실행하고 있는 쓰레드의 'Quantum' 값이나 가장 가까운 타임-아웃값 등등이 될 수 있다. 물론 이렇게 필요한 정보를 미리 설정하는 것이 시스템의 부담으로 작용할 수 있지만, 매 Tick 마다 시간서버 전체가 수행되는 것보다는 적다.

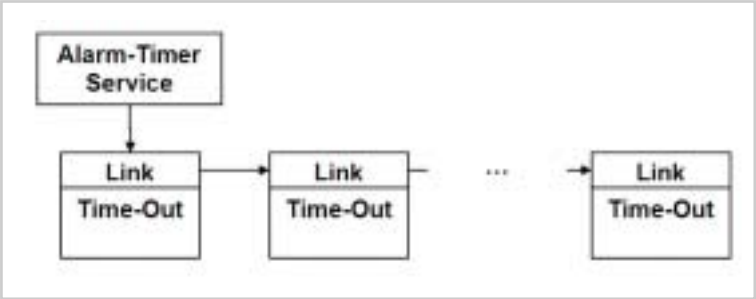

그림 6. Sleep Control 호출도

Tick-Timer management 부분이 일단 수행되면, 시간 서버의 다른 부분들의 모든 서비스를 한번씩 검사하여 필요한 기능을 수행하여 준다. 따라서 시간 서버의 메인시스템이라고 하여도 된다. 다음은 각각의 기능은 아래에서 설명하겠다.

쓰레드 퀀텀 관리

위의 Tick-Timer가 발생하면 현재 수행되고 있는 쓰레드의 퀀텀을 검사한다. 만일 현재 쓰레드의 퀀텀이 모두 소진되어 있다면, 나노커널에서 제공되는 'Task/Thread management' 부분을 호출하게 된다. 나노 커널의 스케줄러가 다음에 어떤 쓰레드를 수행할 지 결정하고, 디스패처(Dispatcher)가 결정된 쓰레드를 수행한다. 퀀텀이 소진되지 않았다면, 아무런 동작도 하지 않는다. 여기에서 중요한 점은 쓰레드 퀀텀 관리 부분에서는 실질적인 동작은 전혀 하지 않고, 나노 커널을 호출할 것인지 판단만 한다. 따라서 매우 적은 시스템 부하가 든다.

슬립 컨트롤

위의 'Tick-Timer'가 발생하면 현재 슬립 상태에 있는 쓰레드의 타임-아웃 값을 검사한다. 만일 쓰레드의 타임-아웃 값이 현재 Tick 값과 비교하여 시간이 지났다면, 해당 쓰레드를 준비 상태로 이동하고, 나노커널에서 제공되는 'Task/Thread management' 부분을 호출하게 된다. 나노 커널의 스케줄러가 다음에 어떤 쓰레드를 수행할 것인지 결정하고, Dispatcher가 결정된 쓰레드를 수행한다. 타임아웃 값이 아직 지나지 않았다면, 아무런 동작도 하지 않는다. 여기에서 중요한 점은 슬립 컨트롤 부분에서는 실질적인 동작은 전혀 하지 않고, 나노 커널을 호출할 것인지 판단만 한다. 따라서 매우 적은 시스템부하가 든다.

시스템-타이머 컨트롤

위의 Tick-Timer가 발생하면 시스템-타이머 컨트롤 부분은 JBOSN RTOS의 다른 서버들에서 설정한 시간값에 대한 서비스를 담당한다. 운영체제에서 필요한 서비스들의 시간관련 정보를 처리하여 해당하는 서비스모듈에 전달하여 준다. 특히, 그림 5에서 보듯이 애플리케이션 타이머 서비스, 드라이버 타이머 서비스 및 알람-타이머 서비스도 이 부분의 제어를 받아서 동작하게 된다.

예를 들면, JBOSN RTOS의 동기화서버의 서비스인 Mutex의 서비스를 제공하는데 타임아웃 기능이 설정되어 있다면, 동기화서버는 시간서버에게 시간관련 설정을 하게 된다. 시스템 타이머 컨트롤 부분은 이러한 시간서비스 요청을 받아서 Tick 값을 기본으로 정해진 서비스시간이 되면, 시간 만료를 동기화서버에 알려준다.

알람-타이머 서비스

알람-타이머 서비스 부분은 JBOSN RTOS의 다른 서버들이나 사용자가 프로그램에서 설정한 시간값에 대해 단 한 번의 알람서비스를 제공한다. 그림 6에서 보듯이, 알람 기능은 설정한 시간이 지나면 설정한 쓰레드에게 메시지버스로 메시지를 전달하여 알람을 전달한다. 대표적인 사용처는 디바이

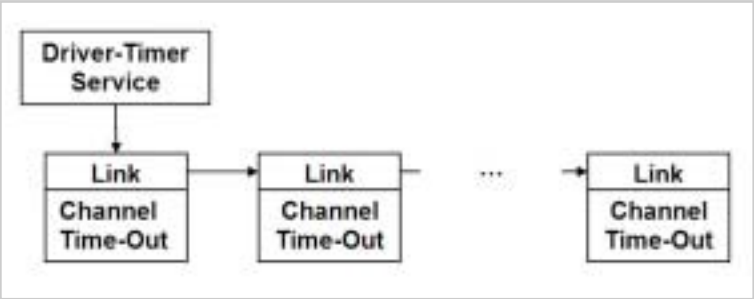

그림 7. Sleep Control 호출도

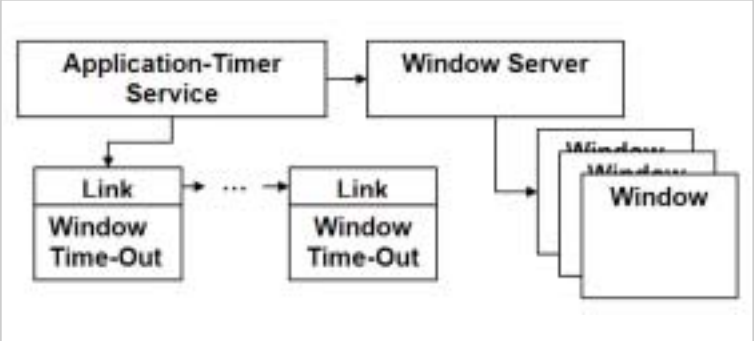

그림 8. Sleep Control 호출도

스 드라이버에 IoRead 함수를 사용하여 데이터를 읽어들일 때, 타임아웃 값을 설정한다고 하자. 이때 타임아웃 값은 알람 타이머를 이용하면 쉽게 구현할 수 있다. 이 부분의 시간 서비스가 특이한 점은 단 한 번의 알람값으로 설정값이 해제된다는 것이다.

드라이버-타이머 서비스

드라이버-타이머 서비스 부분은 JBOSN RTOS의 디바이스드라이버나 서버형태로 제작된 미들웨어에 타이머 서비스를 제공하는 부분이다.

드라이버-타이머 서비스를 요청하면, 반복적이고 지속적인 타이머 서비스가 요청한 서버에 제공이 된다. 그림 7에서 보듯이 드라이버-타이머 서비스 부분은 요청한 쓰레드 에서 할당한 메시지 버스 채널값 및 타임아웃 값을 저장하고 있는 타이머 구조체를 가지고 있다.

이 구조체들은 타임아웃될 순서대로 나열이 되어 있다. Tick-Timer가 발생하면, 이 구조체의 나열을 검사하여 타임아웃된 타이머 구조체에 들어있는 채널에 시그널(SIGNAL) 메시지를 전달한다. 쓰레드는 메시지 버스로부터 시그널을 전달받는다.

애플리케이션-타이머 서비스

애플리케이션-타이머 서비스 부분은 JBOSN RTOS의 응용프로그램에 타이머 서비스를 제공하기 위한 부분이다. 그런데 이 서비스는 그림 8에서 보듯이 응용프로그램에서 직접적으로 시간서버에 요청하지는 않는다. 이 서비스는 윈도우서버에서만 요청이 가능하다. 따라서 응용프로그램 타이머 서비스는 윈도우 'procedure'에서만 요청이 가능하다.

윈도우 procedure에서 애플리케이션 타이머를 요청하면 윈도우서버는 이 요청을 받아서 윈도우 핸들과 타임아웃 값을 이 애플리케이션-타이머 서비스 부분에 윈도우 핸들과 타임아웃 값을 저장하고 타이머 구조체를 생성한다. 이 구조체들은 타임아웃될 순서대로 나열이 되어 있다. Tick-Timer가 발생하면, 이 구조체의 나열을 검사하여 타임아웃된 타이머 구조체에 들어 있는 윈도우핸들을 윈도우서버에 전달한다. 윈도우서버는 이 정보를 받은 다음에 타이머 윈도우메시지를 생성시켜서 윈도우 procedure에 전달한다. Embedded World