J · BOSN RTOS

J · BOSN 실시간 운영체제 포팅 가이드

J · BOSN 실시간 운영체제는 고성능의 멀티-태스킹을 지원하고 매우 강력한 실시간 응용프로그램을 개발할 수 있는 바탕을 제공하여 준다. 특히 운영체제의 기능이 모듈화 되어 있어 사용자가 개발하는 실시간 시스템에 맞게 운영체제를 재구성할 수 가 있다. 이 글은 J · BOSN 실시간 운영체제를 이용하여 포팅하는 방법을 자세히 소개한다.

자료제공 : 아이보슨시스템즈
www.jbosn.com

포팅 가이드(Porting Guide)

구조

J · BOSN 실시간 운영체제는 모듈화를 기본 설계개념으로 하였다. 고성능의 실시간 커널과 다른 모듈을 결합하여 전체시스템이 이루어진다. 커널을 구성하는 모듈도 사용자의 요구조건에 따라 선택적으로 재조립이 될 수도 있다. 모든 모듈은 각자의 독립성을 최대한 보장하고 있다. 각 모듈은 시스템의 서비스를 확장시키는 것이고, 모듈 중 필요한 것들만 결합하여 사용자의 요구조건에 만족하는 실시간 개발 시스템을 제작할 수 있다. J · BOSN 운영체제의 모듈 등은 다음과 같다.

실시간 운영체제인 J · BOSN은 모든 기능이 모듈화 되어 있고 각 모듈은 독립적인 Thread로서 움직이고 있다. 최소한의 기능을 나노커널(NanoKernel)에 구현을 하고 운영체제가 갖추어야 할 필수적인 기능들은 서비스 요청을 받아 서비스를 해주는 서버(server)의 개념으로 작성된 모듈로 이루어져 있다. 따라서 운영체제의 기능은 완벽하게 서로 분리되어 있고, 또한 서로 의존적이지 않다. 이와 같은 구조에서는 필요한 기능만을 사용자가 골라서 자신에게 맞는 최소, 최적의 실시간 시스템을 구현할 수 있도록 하여 준다.

고성능의 실시간 나노커널	사용자의 성공적인 리얼타임 시스템의 구현을 가능하게 해주는 강력한 멀티타스킹 멀티쓰레딩 기능 지원과 서버간의 통신채널의 관리 및 운영체제에 필요한 각종 기능을 최소한으로 구현한 커널이다.
System Server	시스템의 모든 리소스관리를 관리하는 부분이다.
Time Server	시스템의 운영에 관련된 각종 타이머에 관련된 기능을 제공하여 준다.
Synchronization Server	쓰레드간의 동기화 및 통신을 관리하는 부분이다.
I/O System Server	J•BOSN 운영체제는 여러 특성의 디바이스를 접근하는데 동일한 소프트웨어 인터페이스를 제공하고 있다. 사용자는 제공된 인터페이스만을 사용하여 모든 디바이스 드라이버에 접근을 하여 원하는 서비스를 제공받을 수 있다. 또한 리부팅없이 드라이버의 탈착이 가능하고, 다른 모듈과의 의존성을 최소화하였다.
File System Server	실시간기능과 MS Windows와의 호환성을 유지한 매우 빠른 파일시스템이다. 현재는 FAT파일시스템, ROM파일시스템, RAM파일시스템등을 지원하고 있다. 그러나 실제 파일시스템은 사용자의 어플리케이션 영역에 포함이 되어 운영이 될 것이다. 파일서버는 단지 전체적인 파일 관리의 도구를 제공한다. FAT, ROM, RAM파일 시스템은 라이브러리 형태로 제공이 되고 있다.
Window System Server	그래픽을 원하는 시스템에서는 이 서버를 운영하여 기본적인 윈도우 개념을 구현할 수 있으며, 상당히 강력한 GUI를 구현할 수 있다.
Network Server	네트워크를 원하는 시스템에서는 이 서버를 운영하여 네트워크에서 필요한 기본적인 프로토콜 및 소켓인터페이스 개념을 구현할 수 있다.
I/O Drivers	드라이버의 디자인 모델은 특별한 제한조건을 부과하지 않고 사용자와 디바이스의 특성을 고려하여 자유롭게 작성을 할 수 있도록 되어 있다. 제공하는 드라이버 모델은 Stream device driver, Wave device driver, Block device driver와 같다
Board-Support Packages	J•BOSN 운영체제를 개발시스템에 이식을 시키는 부분으로서, 모든 시스템에 대한 동일한 소프트웨어 인터페이스를 제공하여 이식성을 극대화한다. 하드웨어의 초기화와 인터럽트의 관리와 발생, 하드웨어 클락 및 시간 관리, 메모리의 크기와 위치 관리 및 다른 하드웨어 관리를 포함하고 있다.
Boot code	부팅코드는 최초로 Target system에 들어가서 cross development tools에 의해서 개발된 시스템을 Target system에 장착하여 부팅을 할 수 있는 기능을 제공한다.

시작하기 전에

개발시스템 구축

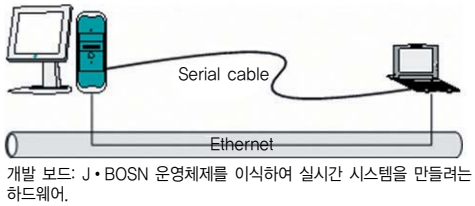
J · BOSN RTOS를 이용하기 위한 최소한의 시스템 환경 설정을 알아본다. 기본적으로 J · BOSN을 이용하여 원하는 시스템을 만들려면 실시간 시스템을 구현할 개발용 하드웨어와 J · BOSN 운영체제를 기반으로 실시간 응용프로그램을 작성하여 컴파일과 링크를 할 수 있는 개발용(Linux나 MS windows) 플랫폼 및 "cross development tools"을 설정하여야 한다. J · BOSN RTOS용 개발용 툴은 www.j · bosn.com에서 다운로드할 수 있다. 기본적인 컴파일러는 GNU-ARM 컴파일러를 사용하고 있다.

J-BUILDER는 J · BOSN RTOS의 하드웨어 이식 및 요구되는 디바이스 드라이버의 개발과 J · BOSN RTOS의 설정(configuration) 및 하드웨어의 설정을 할 수 있는 개발툴이다. 결과물은 J · BOSN RTOS의 최종 이미지이다.

J-STUDIO는 J · BOSN RTOS에 탑재할 수 있는 애플리케이션을 개발하는 기능을 제공하여 준다. J · BOSN RTOS에서 제공하는 각종 라이브러리와 미들웨어를 링크하여 고객이 원하는 애플리케이션을 개발 할 수 있다.

개발 시스템 연결

J · BOSN RTOS를 탑재하기 위해서는 첫 번째로 운영체제를 Loading하여주는 부트로더를 제작하여야 한다. J · BOSN RTOS는 다양한 하드웨어에 이식이 가능한 매우 유연한 운영체제이다. 그림 1과 같이 개발하는데 최소한의 시스템을 기반으로 설명하겠다.



개발 보드: J•BOSN 운영체제를 이식하여 실시간 시스템을 만들려는 하드웨어.
개발용 플랫폼: J•BOSN 운영체제를 기반으로 프로그램을 만들려는 플랫폼으로 개발시스템과 통신기능을 갖추고 있어야 한다. 주로 PC를 이용한다.

그림 1. J•BOSN 운영체제의 간략한 개발 구조

J · BOSN RTOS를 사용하여 실시간시스템을 제작하기 전

에 개발보드와 통신을 위해서 개발용 플랫폼(PC)은 반드시 통신채널을 열어야 한다. 개발용 플랫폼(PC)과 개발보드의 두 시스템은 시리얼을 필수적으로 연결하여야 하고, LAN 연결은 옵션이다. 개발할 때 실시간 시스템의 디버깅은 시리얼을 통하여서 하고, J · BOSN 운영체제를 기반으로 한 실시간 응용프로그램은 시리얼이나 LAN을 통하여 다운로드할 수 있다.

개발용 플랫폼에서 개발된 J · BOSN RTOS 이식결과와 실시간응용프로그램을 다운로드하여 부팅시켜 주는 기본 프로그램으로 부팅롬 프로그램인 "J-BOOT"를 제공하고 있다. 이 프로그램은 최소한의 기본 시스템과 시리얼만을 사용하고 있고, LAN을 연결할 때에는 LAN 인터페이스 소프트웨어를 제공한다. J-BOOT는 시스템의 기본적인 성능 테스트와 메모리 관련 디버깅을 할 수 있으며, J-BUILDER에서 생성된 J · BOSN RTOS이미지 및 애플리케이션을 다운로드할 수 있다. J · BOSN 운영체제는 크기가 매우 작기 때문에 시리얼 통신의 속도에 아무런 영향을 받지 않는다.

시작

J-BOOT 탑재	개발 환경을 설정한 후, J-BOOT를 개발한다. J-BOOT는 개발보드에서 제공하는 툴을 사용하여 탑재가 되어야 한다. 제공된 J-BOOT를 타깃시스템에 장착하여 J · BOSN 운영체제를 이용하여 개발한 실시간 프로그램을 시리얼이나 랜을 이용하여 다운로드하고, 실행, 디버깅을 할 수 있는 기반을 만들어 준다. 이 부팅코드는 J · BOSN의 기본 패키지 안에 들어있다. 하드웨어의 시리얼이나 랜에 관련된 부분은 사용자가 하드웨어 매뉴얼을 보고 직접 수정하여야 한다. 이것을 컴파일하여 개발시스템의 매뉴얼을 참조하여 장착하여야 한다.
케이블 연결	대부분의 개발 보드는 하나 이상의 시리얼을 가지고 있다. 이 시리얼 포트는 부트코드에 의해서 제어되어야 하고, 개발용 플랫폼과 통신을 할 수 있어야 한다. 시리얼 케이블을 연결시키는 방법은 하드웨어 매뉴얼을 참조하기 바란다.
시작	개발플랫폼과 개발 시스템을 정상적으로 설정하였다면, 하드웨어의 전원을 넣고 J · BOSN RTOS를 탑재할 수 있는 준비가 끝났다. 하드웨어에 전원을 넣으면 시리얼을 통하여 개발플랫폼의 시리얼 통신프로그램을 통하여 명령을 기다리게 된다. 사용자는 개발된 J · BOSN RTOS 및 실시간 응용프로그램을 다운로드하여 실행해 볼 수 있다
개발 패키지	J · BOSN RTOS는 인터넷에서 다운로드를 받거나 CD로 제공을 받을 수 있다. J · BOSN 운영체제의 크기는 매우 작아 개발 플랫폼의 디스크를 많이 차지하지 않는다. J · BOSN 운영체제의 커널과 관련된 모든 부분은 라이브러리 형태로 제공되며, 나머지는 소스 형태로 헤더(header)파일, 샘플들이 제공이 된다. 제공되는 패키지를 사용하거나 참조하여 재작성한 후 사용자는 필요한 기능만을 조합하여 원하는 실시간 시스템을 작성할 수 있다.

J · BOSN 패키지 구조

지금부터는 제공되는 J · BOSN 패키지의 구성에 대하여 알아보자.

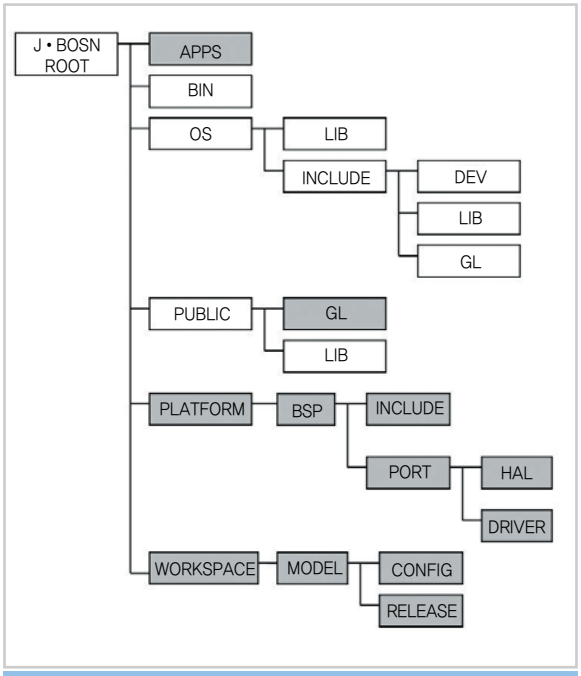


그림 2. J · BOSN 운영체제의 디렉토리 구조

"BIN" 디렉토리는 J · BOSN RTOS를 구축하고 사용하는데 필요한 필수 툴들이 들어 있는 곳이다. 사용자는 반드시 이 디렉토리를 PATH에 추가하여 사용하여야 한다. 예를 들면, 사용자가 개발한 J · BOSN 운영체제의 각 모듈을 하나의 이미지로 만들 수 있는 툴 등 유용한 툴들이 있는 곳이다.

"OS" 디렉토리는 J · BOSN RTOS의 핵심이 라이브로리 형

태로 저장이 되어 있는 곳이다. "OS/INCLUDE" 디렉토리는 J · BOSN 운영체제의 라이브러리에 접근할 수 있는 함수들의 프로토타입과 J · BOSN 운영체제에서 사용하는 기본적인 구조체와 정의들이 들어있다. 또한 기본적인 표준 라이브러리(standard library)는 "OS/INCLUDE/LIB"에 정의가 되어 있다. 여기에 정의되어 있는 것 외에 사용하고 싶다면 사용자가 스스로 첨가를 할 수 있다. "INCLUDE/DEV" 디렉토리는 표준화된 디바이스 드라이버에 관련된 구조체들이 들어 있다.

"PUBLIC" 디렉토리는 공용하여 사용할 수 있는 라이브러리와 샘플코드가 있다. "PUBLIC/GL"은 "window server"를 사용하는 사용자에게 기본적으로 제공되는 그래픽라이브러리이다. 사용자가 자신이 소유한 다른 그래픽라이브러리를 추가할 수 있도록, 소스형태로 제공이 된다.

"PUBLIC" 디렉토리의 다른 것들은 J · BOSN RTOS를 사용하여 응용 프로그램을 작성할 때 참고를 할 수 있는 J · BOSN RTOS의 기능을 사용하는 미들웨어들이 들어 있다.

"PLATFORM" 디렉토리는 J · BOSN RTOS가 탑재되는 시스템의 BSP들이 있는 곳이다. 이 디렉토리에 탑재하려고 하는 시스템의 BSP 코드를 삽입/추가하면 J · BOSN RTOS를 원하는 시스템에 탑재를 할 수 있다.

"PLATFORM/[BSP]/PORT/HAL"은 J · BOSN RTOS를 시스템에 탑재하여 운용하기에 필수적인 하드웨어 관련 부분과 시스템 전체의 메모리 구조 및 시스템 configuration에 관련된 정보를 작성하는 곳이다. 즉, 개발 시스템에 맞게 수정되어야 할 J · BOSN 운영체제의 하드웨어 관련 부분이 존재하고, 시스템의 전체구성(system configuration)을 나타내는 데이터를 J · BOSN 운영체체가 시작할 때 전달한다.

"PLATFORM/[BSP]/PORT/DRIVER"는 드라이버를 작성하는 샘플코드가 들어 있다. 디바디스 드라이버를 작성하는 사용자는 여기를 참조하면 드라이버의 기본적인 작성법을 익힐 수 있다.

"WORKSPACE" 디렉토리는 작성된 PLATFORM에 최종 이미지를 생성하고, 응용프로그램을 결합하여 롬이미지(ROM image)를 만드는 곳이다. "WORKSPACE/CONFIG" 디렉토리는 개발시스템 하드웨어와 J · BOSN 운영체제를 맞추어 주는 코드와 나노커널, 마이크로커널에 해당하는 "time server", "synchronization server" 그리고 "system server", 매크로커널에 해당하는 "IO server", "filesystem server", "window server" 그리고 "network server"를 사용자의 시스템에 맞게 구축할 수 있는 예제가 들어있다. "WORKSPACE/CONFIG" 디렉토리는 사용자가 별도로 작성한 ("APP"디렉토리에서 작성할 수 있다.) 응용프로그램을 추가하여 최종적인 다운로드 이미지를 만들어 낼 수도 있다.

"APP" 디렉토리에는 사용자가 실제로 운영할 프로그램을 작성하는 곳이다. 샘플 프로그램이 제공된다.

J · BOSN 운영체제 이식

이식

이 장에서는 주어진 개발 시스템에 J · BOSN 운영체제를 탑재하는 방법을 살펴본다. 이 장에서 기본적으로 "J · BOSN" 디렉토리에 있는 라이브러리를 이용하여 J · BOSN 운영체제를 타깃 시스템에 탑재하는 방법과 "J · BOSN/PORT" 디렉토리를 작성, 완성하는 방법을 설명한다.

또한 "J · BOSN\PORT\INCLUDE" 디렉토리도 함께 참조가 된다.

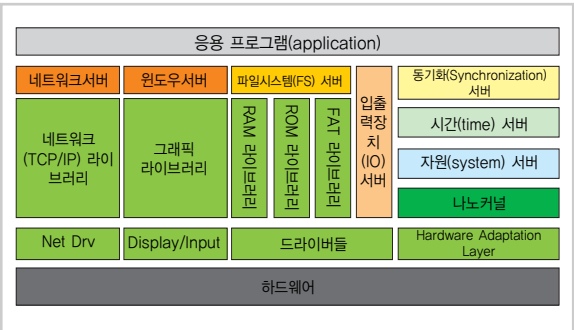


그림 3. J • BOSN 라이브러리 개념도

J · BOSN 시스템 탑재

J · BOSN 운영체제는 "그림 3 J · BOSN 라이브러리 개념도"에서 보듯이 여러 개의 라이브러리로 구성돼 있다. 나노커널과 자원서버, 시간 서버, 동기화 서버로 이루어진 마이크로커널과, 입출력장치 서버, 파일시스템 서버, 윈도우시스템 서버, 네트워크 서버로 이루어진 매크로커널, 그리고 각종 라이브러리와 드라이버, HAL 부분으로 이루어진 하드웨어 관련 플랫폼(PORT)과 마지막으로 응용프로그램으로 이루어져 있다.

이 라이브러리들은 색깔로 구분되어 있는 부분들이 각각 독립적인 이미지를 만들고 여러 개의 이미지를 서로 연결/조합하여 하나의 시스템을 구성할 수 있다. 또는 개발자가 원하는 여러 부분을 하나의 이미지로 만들고 나머지와 연결하여 시스템을 만들 수 있다. 극단적으로 전체 운영체제와 애플리케이션 그리고 하드웨어 관련 부분을 하나로 통합하여 하나의 이미지로 만들어 시스템을 구성할 수 도 있다. 반대로 모든 부분을 각각 다른 이미지(10개)로 만들고 시스템을 구성할 때 하나로 연결시켜주면 각 부분을 시스템 전체로 원하는 부분으로 배치할 수 있다.

모듈화된 J · BOSN 운영체제의 유연성은 시스템 개발자에게 매우 중요한 장점을 제공하게 된다. 시스템을 제작할 때 모든 부분의 중요성이 같지는 않다. 예를 들면, 매우 빠른 실행속도를 요구하면서 시스템에 부담을 주지 않아야 하는 나노커널 부분과 가끔씩 요구되는 그래픽이나 파일시스템 라이브러리들의 중요성은 서로 다르다. 따라서 시스템 제작자들은 나노커널과 같은 매우 중요한 부분을 최고의 속도와 안정성을 갖춘 메모리 영역에 할당하고 비교적 적게 사용되거나 속도에 민감하지 않는 부분은 비교적 싸고 저속의 메모리에 사용하는 것이 전제적인 시스템의 성능 향상에 도움을 준다.

시스템을 작성할 때, 최종적으로 문제가 발생하면 해결하기 위해 디버깅하게 된다. 모듈화되어 있는 구조에서는 디버깅을 각 모듈별로 실행할 수 있기 때문에 빠른 디버깅이 가능하게 된다. 또한 수정 후 테스트 과정이 모듈별로 이루어지므로 개발시간을 단축시킬 수 있다.

모듈화의 가장 중요한 장점은 각각의 모듈이 완벽한 독립성을 확보하고 있으므로 시스템의 안정성이 완벽하게 확보된다는 것이다. 즉, 하나의 모듈에서 발생한 문제점이 전체 시스템의 문제점으로 파급되지 않고, 발생한 모듈에 제한이 되기 때문에 시스템의 치명적인 오작동을 방지할 수 있고, 빠른 수정이 이루어 질 수 있다.

물론 J · BOSN 운영체제는 이와 같은 특성을 최대한 살려 제작되었으며, 모듈에 이상 동작을 스스로 감지하여 모듈의 재부팅을 수행할 수 있는 구조로 되어 있다. 결론적으로 J · BOSN 운영체제는 무한의 안정성을 요구하는 시스템에 최적의 해법을 제시해 준다.

J · BOSN 운영체제 제작 준비

J · BOSN 운영체제를 개발 시스템에 이식을 하기 위해서는 우선 "JBOOT"라는 부팅코드를 개발시스템에 이식해야 한다. 이 코드는 소스 형태로 사용자에게 제공된다. 기본적인 시스템 초기화와 시리얼 통신 포트, LAN 포트를 사용하여 개발 플랫폼인 PC와 연결을 하도록 설계가 되어 있다. 이상의 기본적인 기능을 이용하여 개발된 소프트웨어의 탑재와 실행시켜준다. 또한 기본적인 하드웨어 디버깅을 할 수 있는 기능을 제공한다. 사용자의 요구에 따라서, 제품을 생산할 때나 출시 후 업그레이드기능 및 다양한 개발 시스템의 기능 테스트 항목을 추가할 수 있다.

이 JBOOT에서 작성된 시리얼 통신 포트는 그대로 J · BOSN 운영체제의 콘솔(console)포트로 이용이 될 수 있다. 사용자는 JBOOT의 제작에 사용한 시리얼 통신 프로그램을 그대로 사용하여, 최우선적으로 "PLATFORM/[BSP]/PORT/HAL/console.c"라는 함수를 제작해야 한다.

- PUBLIC VOID ConsoleInit (VOID)
  : 시리얼 통신포트를 초기화한다.
- PUBLIC VOID ConsoleWrite (PBYTE p_Str)

: 시리얼 통신포트를 통하여 주어진 스트링을 출력한다.

- PUBLIC INT32 HAL_DebugReadByte (VOID)
  : 시리얼 통신포트를 통하여 통신내용을 읽는다.
- PUBLIC VOID HAL_HAL_DebugWriteByte (BYTE Ch)
  : 시리얼 통신포트로 주어진 문자를 출력한다.

생성된 "cosole.o"를 다른 모듈에서도 사용할 수 있도록 "J · BOSN"디렉토리로 복사를 한다.

만일, 디버깅이나 panic 등을 사용하지 않을 생각이면 제작을 할 필요는 없다.

J · BOSN 운영체제 기본 모듈

J · BOSN RTOS의 모듈을 탑재하는 방법은 2가지가 있다.

첫째로, "J · BOSN200\os\libram"에 있는 파일을 이용하여 이미지를 만드는 방법이다. 해당하는 디렉토리에는 이미 J · BOSN RTOS의 모듈이 아래와 같이 제공이 되어있어서 따로 모듈을 만들 필요가 없다. JBUILDER를 사용하여 이미지를 만들때 참조되어 삽입된다. 이러한 방법은 기능별로 모

나노커널	"JBOSN200\OS\libram\nkernel.bin"
자원(system) 서버	"JBOSN200\OS\libram\ukernel_system.bin"
시간(time) 서버	"JBOSN200\OS\libram\ukernel_time.bin"
동기화(synch ronization) 서버	"JBOSN200\OS\libram\ukernel_system.bin"
입출력(I/O) 장치서버	"JBOSN200\OS\libram\ mkernel_io.bin"
파일시스템 (F/S) 서버	"JBOSN200\OS\libram\mkernel_fs.bin"
윈도우시스템 서버	"JBOSN200\OS\libram\mkernel_wn.bin"
네트워크 서버	"JBOSN200\OS\libram\mkernel_nw.bin"

듈을 따로 만들어서 보다 안정적인 시스템을 구축할 수 있다.

위와 같은 방법으로 J · BOSN 운영체제의 각각의 모듈을 하나씩 구분하여 제공이 된다.

두 번째는 "J · BOSN200\os\librom\"에 있는 파일을 이용하여 모듈별로 구별하지 않고 작고 빠른 이미지를 만들 때 사용된다. 이 방법을 사용하면, XIP(execution in place) 기능 가능하여, 작은 시스템에서 효율적인 탑재 방법이다. 덧붙여서 응용프로그램도 J · BOSN RTOS와 함께 탑재될 수 있다.