무선 인터넷 데이터링크 레이어의 응답속도를 만족하는 임베디드 시스템 설계

*오현석, 성광수
영남대학교 전자정보공학부
e-mail : ohhs@yumail.ac.kr, 핸드폰 : 010-2802-5703

An Design Of Embedded System for Satisfying Respose Of Wireless Internet Datalink Layer

*Hyun-Seok Oh, Kwang-Soo Sung
School Of Electronic Engineering
Yeungnam University

Abstract

In this paper, we proposed small scale real-time operating system for embedded system. Real-time system is characterized by the severe consequences that result if logical as well as timing correctness properties of system are not met. On real-time system, real-time operating system allows real-time applications to be designed and expanded easily. Functions can be added without requiring major changes to the software. We design small scale real-time operating system for preemptive kernel, and design kernel component such as multitasking, scheduler, task priority, semaphore, inter-task communication, clock tick timer, ISR(Interrupt Service Routine) mechanism has low interrupt latency.

Ⅰ. 서론

고성능 마이크로프로세서와 유무선 통신 기술의 발달, 그리고 소프트웨어 기술의 발달에 따라 실시간 임베디드(embedded) 시스템의 응용이 과거 산업용과 같이 제한된 분야에서 벗어나 정보가전, 휴대폰, PDA, 셋탑박스, 디지털 TV, 자동차 텔리메틱스, 항공기, 산업자동화 등 일상생활로 광범위하게 스며들고 있으며, 한층 더 복잡한 기능을 요구하고 있다. 일반적으로 이러한 실시간 임베디드 시스템은 제한된 메모리 용량이나 전력소모 및 안전과 직결된 실시간성을 요구한다.[1] 이러한 복잡 다양한 기능을 가진 실시간(Real-Time) 임베디드 시스템의 요구에 따라 효율적인 자원관리와 응용 프로그래밍 인터페이스(API : Application Program Interface)를 제공해주는 전용 OS(운영체제)는 필수적인 요소가 되었고, 현재 광범위하게 연구 및 사용이 되고 있다.

특히 최근 PC와 같은 고정된 환경이 아닌 Real-Time OS 기능을 내장한 고가의 임베디드 시스템에서는 멀티태스킹(Multitasking), 파일시스템, 프로세스간의 통신, 많은 아키텍처(Architecture) 지원과 많은 종류의 네트웍 프로토콜 지원을 하기 때문에 OS의 덩치가 큰 것이 단점이다. 만약, 많은 저가의 임베디드 시스템에서 이 Real-Time OS를 적용한다면, 많은 메모리 요구량으로 인해서 저가의 임베디드 시스템에 포팅(Porting)이 불가능할 것이다. 그러므로, 이러한 저가의 임베디드 시스템에서는 덩치가 큰 OS는 적합하지 않다. 그리고, 기존의 Real-Time OS 중에서는 시분할(Time-slice)에 의한 스케줄링(Scheduling)방식을 채택하고 있기 때문에 지금 수행하고 있는 태스크(Task)보다 다음에 수행될 태스크가 우선순위가 높더라도 바로 지금 수행되는 태스크가 중단되는 것이 아니고 할당된 시간이 종료(Time-out)되어야만 현재의 태스크가 CPU를 양보한다.[4] 또한, 현재 수행되는 태스크가 다른 태스크보다 우선순위가 높다고 해서 계속 수행되는 것이 아니라, 역시 할당된 시간이 종료(Time-out)되어야만 다른 태스크에게 CPU를 양보한다. 이런 이유로 인해서, 시분할(Time-slice)에 의한 스케줄링(Scheduling) 방식을 채택한 Real-Time OS는 실시간 기능이 떨어지는 현상이 발생한다. 이러한 실시간 기능을 극복하기 위해 대부분의 Real-Time OS는 태스크의 우선순위를 두고 스케줄링하는 선점형(Preemptive) 스케줄러방식을 사용한다. 하지만, 이러한 Real-Time OS는 고가의 라이센스(License)비로 인하여 저가의 임베디드 시스템에 적용하기가 힘들다.

따라서, Real-Time OS는 덩치가 작아야하고 선점형 Real-Time OS가 되어야 한다. 이러한 소규모 Real-Time OS 설계는 복잡한 일을 처리하는 저가 및 고가의 임베디드 시스템에 많은 도움이 될 것으로 보인다.

이에 본 논문에서는 고가, 저가 임베디드 시스템 모두 적용 가능한 소규모 Real-Time OS를 설계하였다. 이를 8051호환 마이크로컨트롤러인 Dallas사의 DS5002FP에 포팅(Porting)하여 테스트하고, OS의 실시간성을 검증하기 위해서 IEEE 802.11(Wireless LAN) MAC(Medium Access Control)을 응용 소프트웨어로 구현하여 가장 짧은 프레임간 간격(SIFS: Short Inter-Frame Space)을 실시간으로 처리를 만족함을 보이고자 한다.

Ⅱ. 소규모 Real-Time OS 설계

2.1 Real-Time 시스템 개념

실시간 시스템은 시스템이 논리적으로 완벽하게 동작해야 함은 물론이고 정해진 시간에 맞추어 동작하지 않을 경우 심각한 결과가 발생하는 시스템을 말한다. 실시간 시스템은 소프트(Soft)와 하드(Hard) 실시간 시스템으로 분류한다.

2.2 Real-Time OS 구현

마이크로프로세서는 내부 구조에서 고유의 머신 사이클(Machine cycle)을 가지고 있고, 그 짧은 시간을 주기적으로 반복하여 입력된 인터럽트를 체크하며, 선입선처리 방식으로 실행 순서를 정한다. 그러나 한 머신 사이클 내에서의 근소한 입력 시간 차이는 구별 할 수 없기 때문에 이때는 미리 정해진 우선 순위를 적용한다. 이러한 인식하기 힘든 짧은 시간을 감지해야 한다면 마이크로프로세서의 내부 구조적인 오버헤더(Overhead)는 더욱 커질 것이며 그 효율성도 보장할 수가 없다. 또한 실행 중인 인터럽트를 중지하고 발생되는 보다 상위의 인터럽트를 실행하게 하는 네스티드 인터럽트(Nested interrupt) 방식을 채택할 수도 있는데 이는 완전한 선점형 방식으로 낮은 우선순위의 인터럽트에는 응답시간 성능이 저하되는 결과를 초래한다.[5] Real-Time OS의 근간을 이루는 태스크 스케줄러는 대부분 이벤트 구동(Event driven) 방식의 선점형이며 문맥전환(Context switch)시 특정한 우선순위 정책(Priority policy)을 따른다. 그러나, 소프트 Real-Time OS 관점에서 보면 특정한 최상위 태스크만의 완벽한 실시간성을 보장하기 위하여 보다 하위 태스크들의 대기시간(Waiting time)과 응답시간(Response time) 성능을 저하시킬 수는 없다. 본 논문에서는 태스크의 대기 시간과 응답시간 성능을 향상하고 여러 태스크의 병행실행(Concurrent processing)을 고려하여 설계하였고, 이벤트 구동(event driven)형 선점방식을 채택하였다. 선점방식은 시스템의 응답성이 중요한 경우 사용된다. 첫 번째 제안은 한 태스크가 점유하는 CPU 시간을 짧게 하기 위하여 하나의 태스크에 하나의 자원(Resource)만을 할당함에 있다. 이렇게 하여 태스크의 반환(Turnaround)시간 성능이 크게 향상되어 태스크의 실시간성이 향상된다. 두 번째는 현재 실행중인 특정의 태스크는 처리가 완료된 후에도 여분의 CPU 자원을 직접 반환할 수 없도록 제안하였는데, 이렇게 하면 문맥전환이 더욱 더 단순해지기 때문에 오버헤드(Overhead)가 더 많이 줄어든다. 세 번째는 지연(Delay)함수의 기능적 특징이다. 현재 실행중인 태스크는 여러 가지 이유에서 지연함수를 호출하여 실행의 연기를 요청할 경우가 있는데, 지연(예: KTimeDelay();) 함수를 호출하여 현재의 태스크로부터 CPU의 반환을 태스크 스케줄러에게 통보할 수 있다. 이때 태스크 스케줄러는 인자(Parameter)로 넘겨받은 일정시간 동안 현재의 실행 태스크로부터 CPU 자원을 반환하겠다는 뜻을 KTimeDelay() 함수를 통하여 통보받았기 때문에 다음의 타임 슬롯(Time slot) 시작시점이 되면 문맥전환을 실시하고 그 태스크 재시작시간(Resume time)이 경과하기 전까지는 그 태스크를 재실행하지 않도록 스케줄링하여 CPU의 사용률(Utilization) 성능을 향상시킨다.[6] 물론 재시작 시간이 경과하기 전에 해당 이벤트가 발생하면 재실행될 수도 있다.

2.3 태스크와 Real-Time OS의 관계

본 논문에서의 각각의 태스크는 고유의 인덱스(index)를 가지며, 이 숫자는 태스크 고유의 ID에 해당하는 동시에 그 태스크의 우선순위를 의미하며 작을수록 우선순위가 높다. 이 태스크 뒤에는 서로 독립적인 여러 가지의 작업 태스크를 순차적으로 각각의 타임 슬롯에서 실행 시켜주는 태스크 스케줄러가 존재하고 이들의 상호 작용에 의하여 Real-Time OS의 근간이 되는 실시간성과 멀티태스킹(Multitasking)이 보장된다. 또한 태스크 스케줄러는 전용 타이머 인터럽트에 의하여 주기적으로 실행되며 각각의 태스크 실행 시 타이머 관리의 독립성을 보장한다.

태스크는 필요에 의하여 추가 및 삭제가 가능하고 각각의 태스크는 다른 태스크에 영향을 받지 않고 독립적으로 구성될 수 있다. 태스크 스케줄러는 각각 구성된 태스크를 정해진 우선순위에 따라 할당된 타임 슬롯 내에서 차례로 실행시킨다. 문맥전환이 이루어지면 태스크 스케줄러는 정지되는 태스크의 상태를 TCB(Task Control Block)형태로 각 태스크의 스택(Stack) 영역에 보관하여 다시 실행되는 태스크의 TCB는 CPU의 레지스터(Register)로 적재(Load)되어 정지된 직후의 상태를 유지하고 재실행된다.

2.4 Real-Time OS의 구성

본 논문의 Real-Time OS는 기본적으로 인터럽트 핸들러(Interrupt handler), 태스크 스케줄러(Task scheduler), 각각의 태스크, 최하위의 디바이스 드라이버(Device driver)들로 구성되어 있다.

인터럽트 핸들러는 시스템의 근본적인 타임 틱(Time tick)과 이를 근거로 한 범용타이머, 인터럽트, 이벤트 등을 관리하고, 태스크 스케줄러는 각각의 태스크를 우선순위에 따라 실행시켜 주며 이때 각 태스크는 최하위 디바이스 드라이버를 호출하여 입출력 및 자원을 점유할 수 있다. 드라이버(Driver)는 각 주변 디바이스(Device)들과 범용 타이머의 동작에 대한 지원 함수의 형태로 제공된다.

Ⅲ. IEEE 802.11 MAC 소개 및 Real-Time OS을 위한 하드웨어 설계

3.1 IEEE 802.11

정식 명칭이 “IEEE Standard for Wireless LAN Medium Access(MAC) and Physical Layer(PHY) Specification″인 802.11 표준은 근거리에서의 네트워킹을 지원하기 위하여 정의된 무선 통신 프로토콜이다.[7] 802.11 표준은 CSMA/CA(Carrier Sense Multiple Access with Collision Avoidance)를 사용한다. 무선 송수신기는 같은 무선 채널을 사용하여 송신과 수신을 동시에 할 수 없다. 그래서 802.11 무선 LAN은 충돌을 감지하지 않고 단지 피함으로써 스테이션(station)사이에 패킷(packet)을 전송한다.[8]

LLC 계층은 IEEE 802 참조 모델의 최상위 계층이며, 전통적인 HDLC(High-level Data Link Control protocol)로 IEEE 802.11 기반 MAC 링크를 사용하여 최종 사용자간에 데이터를 교환하는데 있다.[9] MAC 계층은 프레임의 인식, 주소지정, 접근 조정, 프레임 체크 시퀀스 생성과 검사 등과 같이 공유매체에 대한 접근 제어 기능을 하는데 있다. PHY 계층은 PLCP(Physical Layer Convergence Procedure)와 PMD(Physical Medium Dependent)의 두 가지 부계층으로 나누어진다. PLCP는 MAC과 무선 전송 프레임을 연결해 주는 가교 역할을 하며, 동기를 맞추기 위한 프리앰블(Preamble)과 자신만의 헤더(header)를 덧붙여 PMD로 넘기는 역할을 담당한다.[9] PMD는 PCLP로부터 넘겨받은 모든 비트를 안테나를 사용하여 공기 중으로 전송을 담당한다.

본 논문에서는 복잡한 변복조를 제공하는 PMD를 구현하지 않고 유선으로 가상 무선을 구현하였다.

3.2 IEEE 802.11 MAC

본 논문에서는 채널에 대한 접근을 예약하기 위해서 RTS(Request To Send) 제어 프레임(Frame)과 CTS(Clear To Send)을 사용하였다. 프레임이 전송되길 원할 때, 수신측으로 데이터 패킷(Packet)과 ACK(Acknowledgement) 패킷의 지속시간을 나타내는 RTS 프레임을 먼저 보내게 되고, CTS 프레임에 대응하는 RTS 프레임을 수신하는 수신측에서는 송신측으로 매체사용을 허락을 위한 CTS을 보낸다. RTS 또는 CTS를 받은 다른 스테이션(Station)들은 데이터 전송이 현재 이루어져 있다는 것을 알게 되고, 전송에 있어서의 간섭을 피할 수 있게 된다.

SIFS(Short Interframe Space)는 RTS/CTS 프레임(Frame)이나 ACK과 같은 최고 우선권을 가진 프레임의 전송을 위하여 사용된다. 고수준의 우선권을 가진 프레임은 SIFS가 지난 후에 통신을 시작할 수 있다.

3.3 Real-Time OS을 위한 하드웨어 설계

본 논문에서 설계한 하드웨어 모듈은 Dallas사의 DS5002FP CPU를 사용했으며, 주된 하드웨어 사양은 다음과 같다.

- PCI Interface
  - PLX9054 (PCI Master, Slave Chip)
- Station
  - DS5002FP 16MHz 8bit (8051호환 Core)
  - 128KByte SRAM
  - FPGA (PLX Interface, Physical Layer Logic)

설계한 Real-Time OS와 응용 프로그램 이미지는 부트로더(Boot-loader)가 내장된 각 스테이션의 DS5002FP에 RS-232C을 이용하여 SRAM에 다운로드하도록 하였으며, 리셋과 동시에 Real-Time OS와 응용 프로그램이 동작하도록 설계하였다. PLX9054를 내장한 PCI 카드는 PC와 내부 스테이션간의 데이터 전송을 담당하도록 설계하였고, 스테이션의 FPGA는 PLX 인터페이스, IEEE 802.11의 가상 물리계층(Physical Layer) 역할과 병렬데이터를 직렬데이터로 변환하는 역할을 하도록 Verilog로 설계하였다. 표 1은 FPGA 내부레지스터(Physical 부분)를 나타냈다.

Name	Width	Access	설명 *(PHY: Physical)*
PERRL	8	RW	PHY Clock Prescale register low byte
PERRH	8	RW	PHY Clock Prescale register High byte
CTR	8	RW	PHY Control register
TXR	8	W	PHY Transmit register
RXR	8	R	PHY Receive register
CR	8	W	PHY Command register
SR	8	R	PHY Status register

표 1 FPGA 내부레지스터 (PHY부분)

전체 동작을 설명하면 다음과 같다.

① PC를 통해 입력된 명령과 데이터는 PCI 카드를 통하여 스테이션1의 FPGA에 보내진다.
② FPGA가 PC로 명령과 데이터를 다 받으면 스테이션1의 로컬(Local) CPU (DS5002FP)로 인터럽트 발생하여 명령과 데이터가 있음을 알린다.
③ 데이터를 받은 스테이션1은 상대 스테이션에게 데이터 보내기 위해, RTS 프레임을 물리계층(Physical Logic)을 내장한 FPGA에게 보내어 매체 사용을 예약한다.
④ FPGA는 스테이션1 CPU로부터 받은 RTS 프레임을 두 선(PCL, PDA Line)을 통하여 스테이션2에 보낸다.
⑤ 스테이션2의 FPGA는 스테이션1로부터 받은 RTS 프레임을 스테이션2 CPU에게 주고, 응답으로 CPU가 준 CTS를 스테이션1로 보낸다.
⑥ 스테이션1의 FPGA는 스테이션2로부터 받은 CTS 프레임을 스테이션1의 CPU에게 주고, CPU가 준 실제 데이터를 스테이션2에 보낸다.
⑦ 스테이션2는 스테이션1로부터 온 데이터를 정상적으로 다 잘 받았다면, ACK 프레임을 스테이션1에 보내고, 최종적으로 스테이션2의 LCD에 데이터를 디스플레이(Display)한다.

Ⅳ. 실험 및 결과

본 논문에서 설계한 Real-Time OS는 하드웨어와 직접적으로 연결되는 일부분만 어셈블리 코드로 코딩했고 나머지는 C 코드로 코딩을 하여 설계하였다. 설계한 Real-Time OS는 Keil사의 컴파일러로 컴파일하고 링크를 수행했다. 결과로 나온 Hex 이미지(OS부분

만)의 크기는 5,482 바이트였다. 본 논문에서는 설계된 Real-Time OS의 성능을 측정하기 위해 스테이션에서 타이머를 사용하여 서비스함수별 평균수행시간을 측정하였다.

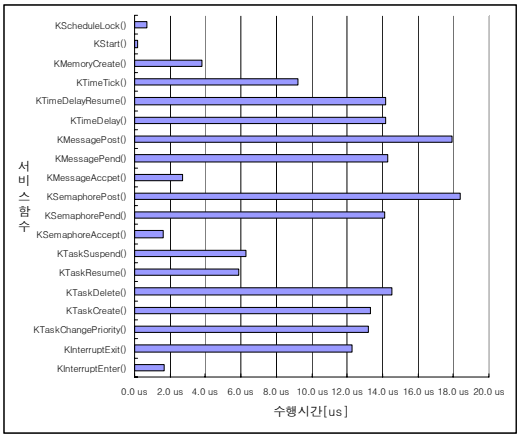

그림 1 서비스함수 평균수행시간 그래프

설계된 Real-Time OS의 실시간성을 검증하기 위해 IEEE 802.11 MAC을 응용 소프트웨어로 구현하여 설계한 OS에서 동작시켰고, 기본적인 실시간을 요구하는 RTS/CTS/DATA/ACK 교환절차를 설계한 하드웨어에서 실험하였다.

각각의 교환절차 사이의 가장 짧은 프레임간 간격(SIFS)값을 측정하여 OFDM(Orthogonal Frequency Division Multiplexing), FHSS(Frequency hopping spread spectrum)방식의 SIFS값과 비교하였다. IEEE 802.11 FHSS방식의 SIFS의 값은 28μs이고 IEEE 802.11a OFDM방식의 SIFS의 값은 16μs이다.[7] 각각의 교환절차에서의 SIFS값은 스펙(Spec: Specification)에서 제안한 값에 실시간으로 응답을 만족해야한다. 설계한 Real-Time OS에서 동작하는 MAC이 FHSS와 OFDM의 SIFS값을 모두 만족한다. 표 3에서, FHSS와 OFDM방식의 MAC은 스펙에서 제안한 값이고, Real-Time OS에서의 MAC은 실험을 통한 결과값이다. 결과값이 스펙에서 제안한 값 이내에 있기 때문에 NOP 명령을 사용하여 지연을 한다면 정확하게 스펙에서 제안한 값 모두에 만족할 것이다.

이에 본 논문에서는 설계한 Real-Time OS가 실시간 처리가 가능하다는 것을 확인하였다.

MAC	시간	비고
IEEE 802.11 FHSS방식의 MAC	28μs (a)	스펙 제안값
IEEE 802.11a OFDM방식의 MAC	16μs (b)	스펙 제안값
설계한 Real-Time OS에서 동작하는 MAC	13.7μs + α	실험을 통한 결과값

(a) FHSS을 만족하기 위한 값 α=14.3μs
(b) OFDM을 만족하기 위한 값 α=2.3μs

표 3 SIFS 제안값과 결과값 비교

V. 결론

본 논문에서는 Real-Time OS를 테스트하기 위한 하드웨어와 FPGA로 가상물리계층을 설계하였고, 임베디드 시스템을 위한 소규모 Real-Time OS를 설계하였다.

설계된 Real-Time OS의 전체 메모리 요구양은 겨우 5,482 바이트로 작은 메모리를 장착한 임베디드 시스템에 저렴한 가격으로 구축이 가능할 것으로 본다.

설계한 Real-Time OS의 실시간성을 검증하기 위해 RTC/CTS/DATA/ACK 교환절차를 실험을 통해 검증하였으며, 검증 결과 설계한 Real-Time OS에서 동작하는 MAC이 FHSS와 OFDM의 SIFS값 이내에 모두 만족함을 확인하였다.

이는 설계한 Real-Time OS가 실시간을 요구하는 저가의 Wireless LAN, 네트웍 임베디드 시스템, 마이크로프로세서를 내장한 모든 응용 임베디드 시스템에 적용가능 할 것으로 본다.

참고문헌

[1] 김선자, 김홍남, 김채규, “인터넷 정보가전용 RTOS 기술 현황”, 한국정보과학회지, Vol. 19, No. 4, 2001. 4.
[2] D. Milojicic and et al., "Embedded Systems", IEEE Concurrency, pp.80 ~ 90, 2000.
[3] C. M. Krishna, and Kang G. Shin, "Real-Time Systems", McGRAW-HILL International Edition, 1997
[4] Jean, J. Labrosse, "uC/OS-II The Real-Time Kernel", R&D Publications, 1999.
[5] 박명순, 김병기, 하순회, 장훈 공저, “컴퓨터 구조 및 설계”, 사이텍미디어사, 1999.
[6] B. W. Kernighan and D. M. Richie, "The C Programming Language", Second Edition, Prentice-Hall, 1988.
[7] IEEE P802.11 D10, Wireless LAN Medium Access Control(MAC) and Physical Layer(PHY) Specifications, Jan., 1999.
[8] 김형훈, “무선 랜 중심의 모바일 통신 기술", *OHM사*, 2002.
[9] 황영상, 김활 공저, “IEEE 802.11을 중심으로한 무선 LAN 바이블”, 세화*사*, 2002.
[10] 매튜 개스트저, 이승철, 강민석역, “802.11무선 네트워크 구축 가이드", 사이텍미디어*사*, 2002.