IEEE-1394 구현을 위한 Embedded LINUX System

서원호*, 이정훈*, 임중규*, 엄기환*
*동국대학교,

Embedded Linux System for IEEE-1394 Realization.

Won-Ho Seo*, jeong-Hun Lee*, joong-Guy Lim*, Ki-Whan Eum*
Dongguk Univ.
E-mail : korean@dongguk.edu

요약

IEEE1394는 내장 리눅스 시스템에 주변 장치를 연결하기 위한 새로운 인터페이스 규격이다.IEEE1394 시스템은 IEEE1394 인터페이스를 가지고 내장 리눅스 시스템과 다수의 주변 기기로 구성 된다. 본 논문에서는 IEEE-1394를 구현하기 위한 Embedded Linux System의 하드웨어적 기반에 대해 기술하고자 한다. 사용한 하드웨어는 MPC860을 탑재한 보드를 사용했으며, Linux kernel은 안정버전인 2.2.14를 사용했고, IEEE1394와 관련된 Application을 탑재하여 이에 대한 성능향상 및 실험을 통해 하드웨어적인 인터페이스를 검증하였다.

ABSTRACT

IEEE1394 is a new interface standard for connecting peripheral devices to embedded linux system. A IEEE1394 system consists of a embedded Linux system and a number of peripherals with IEEE1394 interfaces. In this paper, we Embedded Linux System for IEEE-1394 Realization described with IEEE1394 interface. Using hardware used board on MPC860 processor, Linux kernel used kernel 2.2.14 for a stabilization version, Hardware Interface inspected a test and quality improvement On board with IEEE1394 Application.

I. 서 론

불과 몇 년 전부터 임베이디드 리눅스 커널이 연구용이나 엔지니어 운영체제에서 기업과 사용자들에 크게 부각되고 있다. 이렇듯 리눅스를 선호하는 가장 근본적인 원인은 인터넷이라는 네트워크를 무엇보다도 절실하게 원하는 유저들과 엔지니어 및 프로그래머들의 이해조건이 맞아 떨어짐은 물론, 누구나 자유롭게 접근할 수 있도록 커널 소스가 개방되어 있다는 점이 바로 그 원인으로 작용하고 있다. 이러한 현상 속에서 리눅스는 그 영역을 빠른 속도로 넓혀 가고 있으며, 다양한 형태로 변형을 시키는 연구가 활발히 진행 중이다. 그 중에서도 리눅스를 이용하는 임베이디드 시스템과 RTOS 개발은 리눅스의 커널 소스의 수정을 필수적으로 수반하게 된다. 그러나, 리눅스 커널은 모놀리틱한 구조를 갖는 유닉스계열의 운영체제이기 때문에 각각의 독립적인 기능을 수행하는 블록으로 분리시키거나, 각 블록 내부의 기능을 수정하는 것은 매우 어려운 일이다. 즉 커널 일부분의 변경은 커널 전체의 변경으로 이어지는 것을 의미하는 것이다. 이러한 임베이디드 리눅스 시스템의 특징을 이해하고,분석하여 본 논문에서는 IEEE1394를 구현함에 있어 리눅스 임베이디드 시스템의 하드웨어적인 부분과 IEEE1394의 디바이스에 관련된 내용을 검증하였다.

II.본론

1.Embedded Linux system

임베디드 소프트웨어의 기본은 다양한 CPU에 대한 Cross Compiler 와 다양한 CPU와 하드웨어 구조에 적용될 수 있는 Kernel이다. 이 두 가지 모두 리눅스는 훌륭하게 지원한다. 즉 리눅스는 그 자체만으로도 임베디드 소프트웨어의 기본을 만족한다고 볼 수 있다. 임베디드 리눅스라고 특별하지는 않다. 실시간 기능을 Kernel에 추가한 것을 임베디드 리눅스라고 칭할 수 있겠지만, 임베디드 리눅스에서 실시간 기능이 반드시 필요한 것은 아니다. 임베디드 리눅스를 임베디드 시스템에 리눅스를 탑재할 수 있는 개발 환경을 추가로 제공하는 것이다. 다른 운영 체제와 마찬가지로 일반적인 리눅스 커널은 부팅시에 하드웨어를 초기화하고 메모리상에 자기 자신을 구축하여, 디스크상에 존재하는 파일들과 프로그램들이 커널에서 동작 할 수 있도록 한다. 또한, 하드디스크나 플로피 디스크로부터 부팅되는 일반적인 리눅스 커널과는 달리 Embedded Linux는 EPROM이나 Flash memory와 같은 Non-volatile 메모리에 탑재되어 부팅되게 된다. 따라서 하드디스크를 사용하지 않음으로 인해 ROM이나 RAM을 이용한 파일 시스템을 구성하여 사용하게 된다. 램 디스크 파일 시스템(RAM Disk FS)은 메모리의 일부분을 사용하고자 하는 파티션으로 사용하여 파일을 메모리에 저장하게 되는데, 이는 저장공간을 동적으로 사용할 수 있으며, 이러한 이유로 램 디스크 파일 시스템의 활용도는 거의 100%에 가깝다. 이 구조는 다른 파일 시스템에서와 마찬가지로 디렉토리 구조는 연결 리스트 구조로 되어 있으며, 핸들러에 의해 파일 권한과 보호가 이루어진다.

2. IEEE1394

홈네트워크 분야 중 두 가지의 시리얼 인터페이스 표준이 새로운 인터페이스의 시대를 주도하고 있다. IEEE1394 와 USB가 그것이다. 이중에서 IEEE1394는 애플사가 매킨토시에 사용하기 위해 만든 직렬 인터페이스이다. IEEE1394는 Firewire로 불리기도 하는데, IEEE1394는 각종 주변 기기 케이블을 통합하기 위해서 만들어졌다는 것에서 USB와 기본적인 개발 목적이 일치한다. IEEE1394는 고속의 데이터 전송이 필요한 장비를 목적으로 개발되었으며, USB는 저속의 데이터 전송이 필요한 기기를 목적으로 개발되었다. 현재 사용되고 있는 IEEE1394는 S400규격이 널리 사용되고 있는데, 이 규격에서는 400Mbps의 고속전송이 가능하다. IEEE1394 인터페이스의 가장 큰 특징은 빠른 전송속도에 있다. 모드에 따라 100Mbps, 200Mbps, 400Mbps 등 세 가지 속도를 낼 수 있다. 가령, 400Mbps 속도의 프린터, 그리고 100Mbps 속도의 광디스크가 동일한 IEEE1394 인터페이스 케이블에 물려 있어도 이들은 서로 문제없이 동작하고 서로 데이터를 주고받을 수 있다. 이는 데이터를 주고받을 주변기기끼리 속도를 맞춰가며 상호동작하기 때문이다. 또 다른 큰 특징은 쌍방향 통신 기능이 뛰어나다는 사실이다. 즉 모든 주변 기기마다 IEEE1394를 내장하고 있는 IC를 내장 할 수 있기 때문에 PC등을 통한 화상회의 등의 응용분야에서 성능을 제대로 발휘할 것이다. IEEE1394의 유용성은 무엇보다도 사용 편리성과 속도에 있다. 속도는 규격 상 1.2 Gbit/sec에 이르고 400 Mbit/sec(50Mbyte/sec)는 상용화 되어있다. IEEE 1394는 이론상 6만5천 개의 디바이스를 한 버스에 연결할 수 있다. 그것도 복잡한 세팅절차가 필요 없으며 전원이 들어온 상태에서 플러그를 꽂거나 뺄 수 있으며 이때 자동으로 버스구조가 업데이트 된다(hot plugging). IEEE1394의 내부 전송모드는 크게 비동기 전송(Asynchronous)과 등시 전송(Isochronous)으로 나누어진다. 비동기 모드는 주로 프로그램 데이타와 같이 정확한 전송이 요구되는 경우에 사용이 되고 에러의 검출이나 정정 과정이 동반되며, 아이소크로너스는 비디오 스트림과 같이 실 시간성이 요구되면서도 데이타의 정확성이 중대한 문제가 되지 않는 경우에 사용된다. IEEE1394를 지원하는 디지탈 캠코더가 많이 시판되고 있는데, 이는 주로 아이소크로너스 전송을 사용하게 된다. 반면 하드 디스크나 롬 드라이브류의 보조 기억 장치들은 주로 비동기 전송을 사용한다.

디지탈 캠코더를 비롯한 가전 제품에서는 IEEE1394 대신 i-Link라는 이름을 많이 사용하는데, 본질적으로 동일한 내용이지만, 이는 본래 6선인 케이블에서 전원선 2개를 제외한 4핀 커넥터를 사용한다.

하드디스크, 롬 드라이브와 같은 주변기기들은 기존의 ATA,ATAPI 프로토콜 대신 IEEE1394상에서 SBP-2(Serial Bus Protocol-2)라고 하는 프로토콜을 통하여 커맨드 패킷과 메시지를 주고받는다. 단, 사용되는 커맨드 자체는 SCSI 패킷과 동일하다.

3.개발 환경 및 커널 이식.

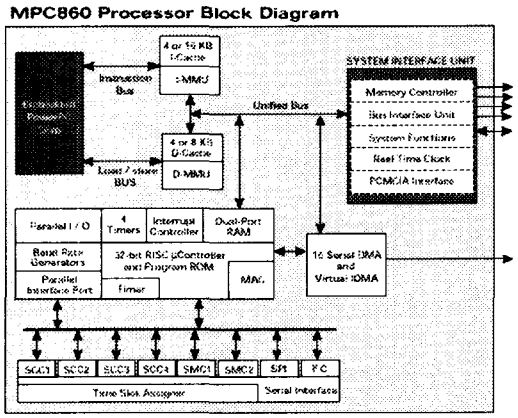
본 논문에서 사용하는 CPU 개발 보드는 고성능 PowerPC 프로세서 코어를 내장하고 있는데, MPC860 프로세서 장치는 다용도의 메모리 제어기 및 독자적인 RISC 마이크로프로세서를 통합하여 분리된 통신 프로세서 모듈을 특징으로 한다. 이 분리된 통신 프로세서 모듈은 내장된 PowerPC 프로세서의 UTOPIA 포트, Ethernet 및 USB를 통한 ATM 트래픽 처리와 같은 주변장치 제어 부담을 덜어 준다.

MPC860 프로세서 장치는 UTOPIA 인터페이스 하나, USB 채널 하나, 2개의 직렬제어기 및 직렬식 주변기기 인터페이스 하나를 포함한다. SCC2 및 SCC3형 제어기는 각각 Ethernet 및 UART를 지원한다. 여기서는 USB 채널을 IEEE1394 채널로 변경하여 임베디드 리눅스를 구현 하고자 한다. MPC860 개발 보드 및 IEEE1394의 특징은 다음과 같다.

▶860 target board
-CPU : MPC860DSL(50MHz)
-ROM : 2MByte Flash memory
-SDRAM : 16MByte
-Ethernet 2 port, Serial Port, BDM port
-10 BaseTx
-BDM port를 이용한 Mux-ice

▶IEEE1394 Controller
-Physical Layer: TSB21LV03C(TI사)
-Linklayer Layer: TSB12LV21B(TI사)

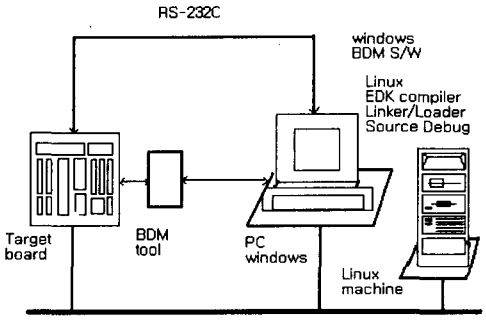
[그림1]Block Diagram



3.1개발환경

MPC860은 PowerPC RISC processor, Memory Controller, Interrupt Controller, Communication 용 RISC processor등을 내장하고 있는 SOC(System On Chip) 타입의 임베디드 프로세서이다. SMC1을 이용한 9600 baud rate Serial과 SCC1, SCC2를 이용한 2 Channel 10Base-T Ethernet, 그리고 Burst read/write 기능 확인을 위해서 IDMA 기능을 구현하였다. IEEE1394 구현을 위한 임베이디드 리눅스 개발을 위해 사용된 장비 및 개발환경은 [그림 1]과 같다. BDM tool인 visionICE는 MPC860이 탑재 된 Target과 10pin BDM connector로 연결되고, Host와는 Parrallel port로 연결된다. Target과 Host는 RS-232 cable로 연결되어 Character 입출력을 하게 되고, Host에 설치된 Ethernet packet analyzer program인 NetXRay를 이용하여 Target의 Ethernet packet 송수신 처리 기능을 검증하였다. visionICE는 Host에 설치되는 visionClick, visionSIM, visionUtility를 통해서 Target의 MPC860 레지스터 초기화, 소스레벨 디버깅 및 Flash memory fusing 기능등을 지원한다.

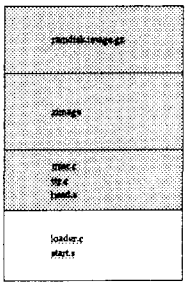
[그림 2] 개발 환경.



3.2 커널 초기화 구조

PC에서의 동작순서는 매우 복잡하다. 복잡한 이유는 하드웨어가 복잡하고 Disk Driver에서 부팅이 되기 때문이다. 즉 리눅스가 복잡한 것은 아니다. 기본 하드웨어에서 리눅스가 수행되기까지 과정은 먼저 전원이 공급되면, CPU에 Clock이 들어가고, 그러면, CPU에 Reset 신호가 풀린다. 그런 후 CPU는 ROM으로부터 데이터(명령어)를 읽어 온다.

[그림3]ROM 저장 데이타



ROM에 있는 명령어를 통해 아래와 같이 하드웨어를 초기화 해야 한다.

-CPU 내부 Cache가 있는 경우 cache을 초기화 한다.(start.s)
-CPU에서 MMU(Memory Management Unit) 기능을 제공할 경우 MMU을 초기화한다.
-메모리 제어기를 초기화 한다. 이제부터 DRAM에 데이터를 쓰고 읽을 수 있다. DRAM의 시작 번지는 0이다. (start.s)
-ROM에 있는 리눅스 관련 이미지(start.s 제외한 나머지)를 DRAM에 복사한 후 이미지 시작 번지로 간다.(loader.c)
-RS-232 Device(UART) 초기화 한다.(head.s)
-리눅스의 압축된 커널을 푼다.(tty.c, misc.c)
-커널을 시작한다.
-압축된 파일 시스템(RAM disk Image)을 푼다.
-RAM Disk을 생성한 후 푼 파일 시스템을 RAM disk에 복사한다.
-RAM disk에 있는 시작 파일(/sbin/init)을 수행한다.

그림3은 ROM에 있는 내용을 보여준다. 여기서 start.s와 loader.c는 개발자가 작성해 주어야 한는 부분이다. start.s를 수행하면 하드웨어는 DRAM과 UART를 읽고 쓸 수 있게 된다. loader.c는 ROM에 있는 linux srouce부터 생성된 이미지를 ROM에서 읽어서 DRAM에 복사한다. head.s는 C 프로그램인 serial.c와 misc를 수행하기 위한 준비 작업(stack pointer와 특정 메모리 영역 정의)을 한다. tty.c는 UART를 초기화 하고 Misc.c는 압축된 커널을 푼다. head.s, tty.c는 하드웨어 구조에 따라 다르게 작성된다. 따라서 이프로그램도 하드웨어를 잘알고 잇는 사람이 작성해야 하는 부분이다. 리눅스 커널을 소스(/user/src/linux)에서 arch/ppc를 들어가서 보면 무슨 무슨 boot라는 디렉토리가 여러개 있다 .Boot 앞에 단어가 의미하는 것은 특정 상용 하드웨어를 의미한다. 해당 디렉토리 밑에서 일반적으로 널리 알려진 사용 하드웨어에 대한 head.s, tty.c, misc.c의 source가 있다. 이를 참조해서 해당 임베디드 시스템에 맞는 init.s, tty.c, misc.c을 작성하면 큰 도움이 될 수 있다.

리눅스 커널에서 하드웨어 종속적인 부분에 대한 소스의 변경이 필요하게 되며, 이는 리눅스 커널이 하드웨어 플랫폼별로 분류를 하고 있어 편리하게 되어 있다. 본 논문을 위한 작업에서는 arch/ppc 디렉터리를 주로 변경하여 사용했으며, 초기화에서 하드웨어에 관련된 부분은 어셈블러와 C 언어를 공통으로 사용하였다.

MPC860을 초기화하기 위한 초기화 코드는 일반적으로 EPROM이나 FlashROM에서 수행되며, 이식 작업 초기에는 RAM으로 download 하여 수행하여 debugging 후 ROM화를 한다.

MPC860에 전원이 인가되면 기본적으로 CS0으로 지정된 번지를 액세스하여 코드를 실행하기 시작한다. MPC860에서 초기화가 필요한 부분은 다음과 같다.

-PowerPC core registers
-Memory Management Unit(MMU)
-Instruction 및 Data Cache
-Clock circuitry
-System Interface Unit(SIU)
-User Program Machines(UPMA)
-Chip Select Machine
-UPM Tables
-Baud Rate Generator
-SMC

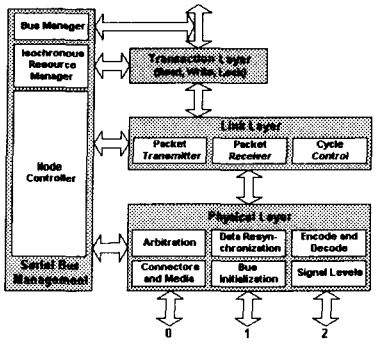
레지스터들의 초기화에서, 우선 초기 값을 설정해 주어야 하는 레지스터는 HRCW(Hard Reset Configuration Word) Power-on Reset 과정에서 HRCW가 데이터 버스로부터 읽혀지거나 이것의 default 값이 0인 경우에 RSTCONF 값이 결정된다. 다음으로는 PowerPC Core register 중 하나인 MSR(Machine State Register)가 초기화되는데, 이는 기본적인 시스템 기능을 규정하는 것으로 default 값은 0이다.

HRCW에 있는 IP 비트는 소프트웨어적으로 reset 과정을 하는 데 있어서 중요한 역할을 한다. 이것은 Reset 발생 시 PowerPC Exception Vector Table이 어디에 위치할 것인가를 결정하게 된다. 만일 IP가 0(default)이면 MSRip 비트 25에 기록된 값은 1이 되는데, 이 경우 Exception Vector Table은 주소 0x000n_nnnn에서 시작되게 된다. HRCW에 있는 Initial ISB(Internal Space Base)비트 역시 중요한데, 이는 IMMR(Internal Memory Map Register)의 값을 결정하기 때문이다. IMMR 값은 내부 메모리 맵 레지스터 공간의 시작 주소를 가리키고 있다.

3.3 IEEE1394의 프로토콜

IEEE1394는 세가지 프로토콜 Layer가 있다. Serial Bus Management는 Physical Layer, Link Layer, Transaction Layer라는 3가지의 Layer와 연결되어 있다. Physical Layer는 IEEE1394 커넥터와 연결되어 있고, 다른 Layer는 애플리케이션과 연결되어 있다.

[그림4]IEEE1394 프로토콜



Physical Layer는 IEEE1394 디바이스와 케이블 사이에 전기적, 물리적으로 연결되어 있으며, 실제 데이터를 송,수신하며 모든 디바이스가 버스를 순차적으로 실행하고 각 포트에 동일한 기능을 제공하는 Repeater의 역할도 한다.

Link Layer는 비동기와 등시 전송 패킷을 송수신하기 위해 두 개의 FIFO와 한 개의 수신 FIFO를 가지며, 각 FIFO는 32비트의 길이로 사용자가 FIFO의 크기를 소프트웨어로 결정할 수 있다. 송신 전용인 비동기용 FIFO와 등시용 FIFO는 write용으로, 수신 전용인 FIFO는 read용으로 사용된다. 비동기 전송은 데이터와 계층 정보가 명시된 어드레스로 전송하며, 프린터나 스캐너처럼 실시간으로 동작하지 않아도 되는 정보를 전송할 때 사용한다. 등시전송은 데이터를 보낼 때 어드레스를 사용하지 않고 , 채널번호를 포함시켜서 전송한다. 즉, 실시간 전송을 하기위해 에러가 나더라도 재전송을 요구하지 않는다. 이런 등시전송은 동화상이나 음성 정보처럼 시간적인 제약이 많은 멀티미디어 정보를 전송할 때 사용된다.

Transaction Layer는 비동기 프로토콜의 read, write, lock기능을 한다. write는 송신 측에서 수신 측으로 데이터를 보내고 read는 데이터를 송신 측으로 보낸다. lock은 write와 read명령의 조합기능으로 수신 측과 송신 측 사이가 현재 통신중일 경우 다른 송신 측의 앞의 통신이 다 끝난 후, 재송신하는 기능이다.

Serial Bus Management는 타이밍 조정과 버스에 있는 모든 디바이스에 전원공급, 모든 시리얼 버스를 관리하며, 사이클 마스터 , 등시 ID, 오류인식 등의 역할을 각 layer에 부여한다. 버스 매니지먼트는 IEEE1212 표준 레지스터 구조로 만들어졌다.

이러한 프로토콜들에 의해 IEEE1394 인터페이스는 동작하는 상황에서 새로운 주변기기가 네트워크에 추가되거나, 혹은 기존에 사용되고 있던 장치가 네트워크로부터 떨어져 나갔을 때는 네트워크의 구성이 재조정되고, 이때 네트워크에서 전송이 이루어지고 있던 모든 기존 정보는 초기화되고, 전체 네트워크는 동적으로 재구성되며 각각의 노드는 어드레스를 다시 부여 받는다. 이 경우 루트 노드도 필요하다면 강제로 가장 많이 사용되는 노드를 루트로 지정할 수도 있다. 그런 다음 루트 노드의 구성이 끝나면 자체 인식 차례가 되어 각 노드들은 네트워크 전체에 걸쳐서 자신의 존재를 다른 노드에게 알려 준다. 이런 식으로 모든 노드의 정보가 수집된 다음 IEEE1394 인터페이스는 정상동작을 시작하기 위한 대기상태로 들어가는 것이다.

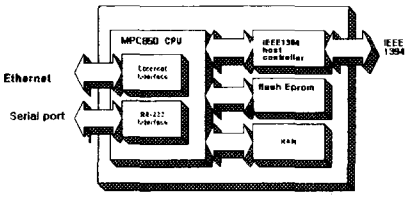
3.3 IEEE-1394를 위한 I/O 포트 탑재 및 설계.

일반적으로 하드웨어 관련 부분에 대한 초기화 작업이 완료되고, 목적에 맞는 입출력 Device driver를 완성한 후 응용 프로그램을 이식하는 절차를 거치게 된다. 우선 직렬포트인 SMC1을 사용하여 콘솔기능을 대신하도록 하여 부팅 시 표시되는 화면 출력을 볼 수 있도록 하였다. 그리고, SCC2와 Scc3를 각각 사용하여 포트 1과 2로 사용하였다. 그리고, 나머지 SCC1을 IEEE1394 포트로 사용하였다.

IEEE394 포트에 연결된 IEEE1394 controller에 link layer는 TI사의 TSB12LV21B controller로 이 디바이스는 PCI-Lynx를 지원하고 32-bit PCI I/F와 4k FIFOs를 가진 PCI to 1394로 전원 3.3V로 지원되고, physical layer는 TI사의 TSB21LV03C controller 디바이스로 TEEE1394-1995, 3.3V, 3port, 100/200Mbps를 지원한다. 이들 디바이스는 [그림5]에서 IEEE1394 Host Controller가 그 역할을 담당하게 될 것이다.

MPC860 core에 관련된 하드웨어 인터데이스 부분은 [그림5]와 같이 설계된다.

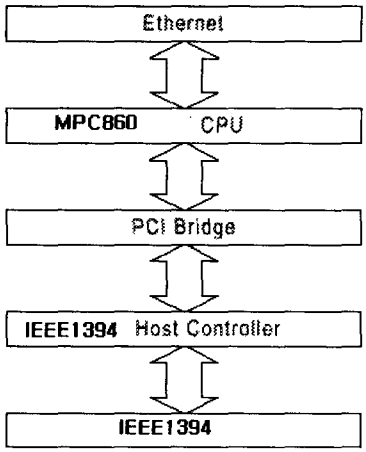
[그림5] TI사 controller Used hardware design



TI사에서 지원하는 Link layer controller는 PCI-Lynx의 기능을 가지고 있다.

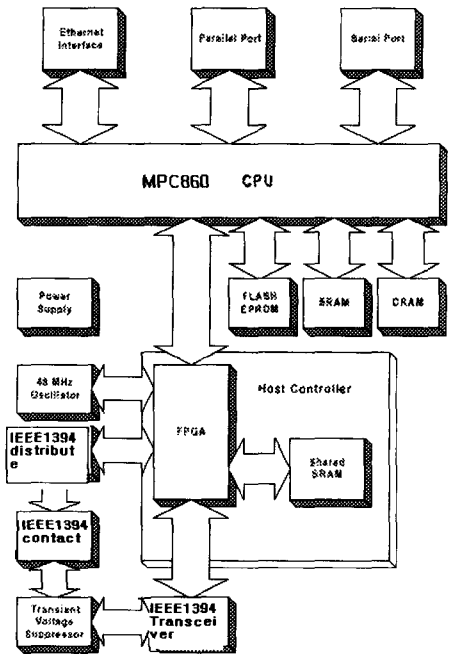
IEEE1394 Host controller는 PCI bus와 연결된다. 이 PCI bus에 MPC860 core가 연결되는데, 다음 [그림6]은 이를 잘 보여 주고 있다..

[그림6]Connnecting a PCI based USB Host controller



다음은 MPC860 Hardware의 block diagram을 나타내고 있는데, 이를 기반으로 MPC860 cpu와 IEEE1394 controller간의 인터페이스를 디자인 하게 된다. 또한 이에 관계된 주변 디바이스와의 상관관계도 [그림6]에서와 같이 설계되어진다.

[그림6]MPC860 Hardware Block diagram



4. 결 론

본 논문에서는 요즘 부각되고 있는 IEEE-1394를 사용하기 위한 Embedded Linux System의 하드웨어적 환경을 살펴 보았다.이 IEEE 1394 표준 인터페이스는 현재 Compaq의 Presario 시리즈 등에서 채택하고 Windows 98에서 구동 소프트웨어를 지원하는 등의 대중화를 위한 노력이 계속되고 있지만, 아직 시장에서의 역할여부가 불확실한 상황이다. 그러나, IEEE 1394는 현재 PC에서의 당면한 입출력 전송속도 문제를 해결할 수 있는 유일한 대안이며, 최근 DVD를 포함한 가전기기에서 이를 채택하여 댁내망(Home Network)을 구성함에 있어 크게 관심을 불러 모으고 있다. 이러한 시점에 본 논문에서는 IEEE-1394를 구현하기 위한 Embedded Linux System의 하드웨어적 기반에 대해 IEEE1394 인터페이스 구현을 시도 하였다. 하드웨어적으로 MPC860을 탑재한 보드를 사용하여 설계하였고, Linux kernel은 안정버전인 2.2.14를 사용했으며, IEEE1394와 인터페이를 구현하여 성능향상 및 설계 그리고, 테스트를 통해 하드웨어적인 인터페이스를 검증하였다, 그결과 양호한 IEEE1394 port를 위한 인터페이스가 가능함을 알 수 있었다.

이를 기반으로 하여 향후 시스템의 안정성과 성능향상을 위한 연구개발 작업을 지속적으로 진행시켜 보다 효율적인 시스템을 제공할 것이다.

참고 문헌

[1] D.Anderson,"FireWireSystem Architecture: IEEE1394", Addison-Wesley, 1998
[2] Philips, "Digital Interface for Consumer Electronic Audio/Video Equipment", Draft v2.1, October 1995
[3] IEEE 1394-1995, Std for High Performance Serial Bus, 1995
[4] MOTOROLA "MPC860 user's manual power QUICC II"/MPC860UM/AD 07/98 REV.1
[5] Daniel Povet& Mtcocesati "understanding the LINUX KERNEL" , January 2001
[6] Alessandro Rubini "LINUX DEVICE DRIVERS" , February 1998
[7] Addison-Wesley "LINUX KERNEL I NTERNALS". second edition 1999
[8] URL : http://www.ti.com/src/1394/
[9] URL : http://www.e-motora.com/
[10] Patrik Persson and Lars Selander, "An Internet Camera Server for USB Cameras", August 29,1997