HDD를 이용한 저장·재생기의 구현 및 디바이스 드라이버 프로그래밍

최효정, 이중호, 김대진
전남대학교 공과대학 전자정보통신공학과

The Implementation of Recording and Replaying System and Its Device Driver Programming

Hyo-Jung Choi, Joong-Ho Lee, and Dae Jin Kim
Electronics Engineering Department
Chonnam National University
E-mail : u0220531@moiza.chonnam.ac.kr

Abstract

Introduction of digital broadcasting service does not only mean the change of information transmission method but also the change of total broadcasting system. In past day, Television was only received one-sided information from broadcasting station, but digital broadcasting means that digital television becomes the most important means of information transmission by the introduction of new programming, lots of channels, data service, multi communication.

In the age of the digital broadcasting, the recording and replay medium's interest is getting higher. The medium is able to record more than 24 hours' digital broadcasting programs without additional tapes.

In this paper the recording and replay device using HDD was implemented and device driver based on linux was programmed. It has Intel PXA250 processor and hard disk is used as storage equipment. And transport Stream is saved on hard disk through PXA250's data bus. FIFO is added to solve the different saving speed and FPGA is also added to display the saved data.

I. 서론

통신과 컴퓨터 및 방송이 융합 되어 멀티미디어화 함에 따라 기존의 아날로그 방식의 방송이 디지털화 되고 있다. 특히 미국, 유럽, 일본 등 선진국은 이미 위성을 통한 디지털 방송을 일부에서 실시하고 있으며 디지털 지상파 방송을 2000 년 이전에 실시하기 위한 계획을 발표하였고 오래 전부터 많은 투자와 연구를 수행해 왔다.

우리나라의 경우는 1996 년에 정부의 지상파 방송 디지털화 계획이 발표되고, 1997 년 11 월에 미국 방식이 선정되었으며, 2000 년에 시험 방송을 실시하고 본 방송은 수도권지역은 2002 년까지 광역시 권은 2003 년까지 도 권은 2004 년까지 시군 권은 2005 년까지 본 방송을 실시한다는 계획을 진행중이다.

디지털 텔레비전이 단순한 텔레비전의 의미를 떠나 정보가전기기로 떠오르면서 텔레비전으로부터 얻을 수 있는 정보량 막대하며 이러한 정보의 효용 가치는 매우 클 것으로 예상된다. 이렇게 디지털 텔레비전으로부터 얻을 수 있는 유용한 정보를 저장하여 시청자가 원할 때 다시 활용 하기 위해서 PVR 과 같은 저장매체가 필요하다.

본 논문에서는 이렇게 수요가 증가하고 개발이 활발히 진행중인 PVR 기능을 할 수 있는 HDD 를 이용한 저장·재생기를 구현하고 리눅스 디바이스 드라이버를 프로그래밍 하였다. PVR 뿐 아니라 저장매체를 필요로 하는 정보가전에 HDD 를 손쉽게 응용하고 활용할 수 있도록 임베디드 시스템의 형태로 HDD 를 장착 하였고 다양한 기능을 위해서 PXA250 를 MCU 로 사용하였으

한국과학재단 지정 전남대학교 고품질 전기전자부품 및 시스템 연구 센터의 연구비 지원에 의해 연구되었음.

며 리눅스를 OS 로 탑재하였다.

II. HDD 를 이용한 저장·재생기의 구조

2.1 Xscale(PXA250)프로세서부의 구조

PXA250 은 인텔사의 Xscale 시리즈 중 하나이며 ARM 에 기반하여 Intel 사에 자신의 아키텍처를 첨가한 구조로 기존의 ARM 과 호환성이 있다. PXA250 은 저전력 고성능 CPU 이며 다양한 주변 장치와 호환이 가능하다.

PXA250 은 32 비트 프로세서로 메모리 맵을 가지고 있다. 전체 4Gbyte 의 메모리 맵을 가지고 있으며 static 메모리 영역과 PCMCIA 메모리 영역, 내장된 디바이스 레지스터 영역, SDRAM 메모리 영역, 그리고 예약영역으로 나뉜다. [1]

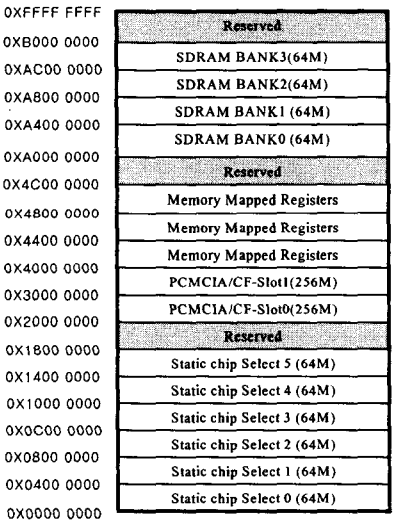

그림 1. PXA250 의 메모리 맵

프로세서 주변 장치로는 부트로더와 커널이미지를 올릴 수 있는 플래쉬 메모리(Static Chip Select 1)와 SDRM 을 메모리(SDRAM BANK 0)로 사용하였고 이더넷 지원을 위한 이더넷 컨트롤러는 Cyrus Logic 에서 개발된 10Mbps 를 지원하는 CS8900(Static Chip Select 0)을 사용하였으며 IDE 인터페이스를 이용하여 데이터를 저장할 수 있는 HDD(Static Chip Select 3)를 장착하였다.

2.2 저장부의 구조

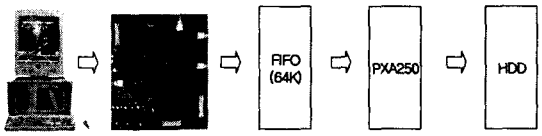

그림 2. 저장부의 세부 블록도

트랜스포트 스트림을 PXA250 의 데이터 버스를 이용하여 저장하기 위해서 PXA250 과 하드디스크 드라이버를 설계하고 PXA250 과 트랜스포트 발생기 부분을 설계하였다

우선 PXA250 과 하드디스크 드라이버의 연결을 위해서 IDE 인터페이스를 사용하였다. IDE(Integrated Drive Electronics)는 컴퓨터의 디스크와 저장 장치간에 사용되는 표준 전자 인터페이스이다.

IDE 하드웨어의 설계는 PXA250 과 하드디스크 드라이브의 동작 전압의 차이를 해결하기 위한 전압 레벨 전환 회로를 추가 하였고, 리눅스의 커널상에서 메모리를 할당 시켜 주고 커널이 인식 할 수 있도록 일부 커널을 수정하는 작업을 하였다.

PXA250 의 읽고 쓰는 속도가 트랜스포트 스트림의 속도와는 다르기 때문에 효율적인 저장을 위해서 추가적으로 FIFO 를 설계하였다. FIFO 는 64K 정도의 용량을 가지며 read 와 write 가 독립적으로 이루어진다. FIFO 에 저장된 트랜스포트 스트림은 FIFO 가 일정한 용량이 채워졌을 때 인터럽트를 사용하여 PXA250 의 데이터 버스로 입력되도록 설계하였다.

2.3 재생부의 구조

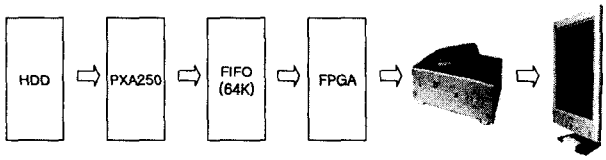

그림 3. 재생부의 세부 블록도

하드디스크 드라이브에 저장된 트랜스포트 스트림은 PXA250 데이터 버스로 읽혀지게 된다. 출력된 데이터는 저장부와 같이 FIFO 에 일시 저장된다. 원래 스트림 발생기에서 발생된 MPEG-2 트랜스포트 스트림 패킷은 8 비트의 데이터, 1 비트의 DVALID, 1 비트의 PSYNC, 1 비트의 CLK 으로 이루어져 있는데 하드디스크 드라이브에 저장은 8 비트의 데이터만 하기 때문에 재생을 위

해서는 완전한 MPEG-2 트랜스포트 스트림을 만들어 주어야 한다. 이는 트랜스포트 스트림의 DATA 가 188 비트마다 '47H'이라는 데이터가 규칙적으로 나타나는 성질을 이용하게 되는데 이 규칙을 이용하여 데이터 8 비트에 DVALID, PSYNC, CLK 성분을 추가해 준다.

이를 위해서 10K 계열의 EPF10K10TC144 로 FPGA 를 구현하였다. FPGA 는 MAXPLUS-II 툴을 사용하여 VHDL 로 코딩 하여 구현하였다.

III.부트로더와 디바이스 드라이버의 구현

3.1 개발 환경 구축

리눅스 디바이스 드라이버를 개발 하기 위해서는 우선 컴파일 환경 구축이 필요하다. PC 에서 동작되는 프로그램은 PC 에서 동작되는 컴파일러를 이용하지만 PXA250 과 같은 임베디드 시스템의 경우에는 이러한 작업이 불가능하므로 크로스 컴파일 환경이 필요하다.

크로스 컴파일 환경이란 호스트 시스템에서 구현하고자 하는 임베디드 시스템용 리눅스를 개발하기 위한 환경을 말한다. 먼저 해당 프로세서에 해당하는 툴 체인 환경을 구축하여야 한다. 툴 체인은 각종 프로그램 소스들을 컴파일하고 빌드하여 실행 가능한 바이너리를 생성하는 데 필요한 각종 유틸리티 및 라이브러리를 말한다. 기본적으로 어셈블러, 링커, C 컴파일러, C 라이브러리 등으로 구성되어 있다. [4]

3.2 부트로더

일반적으로 부트로더는 일반 i386 리눅스에서 LILO 를 많이 사용한다. LILO 란 Linux Loader 로써 도스나 NT, 리눅스 등 다른 OS 를 선택적으로 부팅 할 수 있도록 하는 기능을 제공한다.[4]

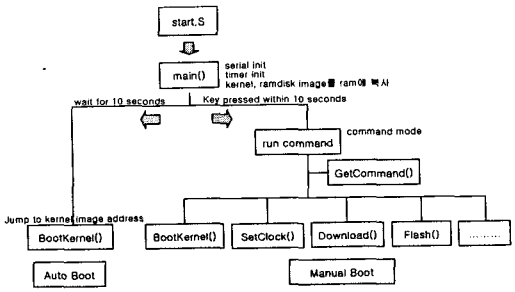

그림 4. 부트로더 흐름도

부트로더에서 가장 먼저 실행되는 파일은 Start.S 란 파일로 이 파일을 통해 CPU speed, 메모리, 인터럽트, UART 등을 초기화 한다. 부트로더의 가장 중요한 기능은 Kernel 이미지나 ramdisk 이미지를 다운로딩 하는 기능이다. Host 상에서 컴파일된 이미지를 시리얼이나 tftp 를 이용하여 이더넷을 통해 SDRAM 상으로 다운로딩이 가능하다. 다운로딩한 Kernel 이미지와 ramdisk 이미지는 SDRAM 상에 있기 때문에 전원이 꺼지면 다운로딩한 이미지는 지워져 버린다. 그러므로 플래쉬 기능을 통하여 SDRAM 상의 커널과 ramdisk 를 지정된 플래쉬 메모리 주소 영역에 쓰는 기능을 한다.

부트로더의 처음 시작은 Start.S 에서 인터럽트 초기화, CPU 속도 설정, 메모리 설정을 한다. 초기화 작업 후 사용자로부터 입력을 기다리게 되며 10 초 동안 입력이 없을 경우에는 커널로 점프하게 된다. 만약 10 초 내에 사용자로부터 키 입력을 받게 되면 커널 이미지나 ramdisk 이미지를 플래쉬 메모리에 쓰는 기능을 하게 된다.

3.3 디바이스 드라이버 구현

시스템에 있는 각각의 주변 장치들은 그 장치들을 제어하는 디바이스 드라이버를 가지고 있다. 이러한 디바이스 드라이버는 실제 장치를 제어하는 부분을 추상화 시켜 서로 다른 종류의 디바이스들을 정형화된 인터페이스를 통해 사용자 프로그램이 이용 할 수 있는 방법을 제공해준다. 다시 말해서 디바이스 드라이버는 디바이스를 관리 할 수 있는 정형화된 인터페이스를 구현하기 위한 함수와 자료 구조의 집합이라고 할 수 있다. 커널은 이렇게 구현된 인터페이스를 통해 물리적인 디바이스에게 I/O 연산을 요청하고 제어할 수 있다.

재생과 저장을 위한 디바이스 드라이버는 모듈형태로 제작되었다. 모듈이란 하나의 오브젝트 파일(*.o)으로서 실행중인 커널에 동적으로 적재하거나 제거될 수 있는 프로그램으로 순차적인 프로그램이 아닌 이벤트 처리 프로그램의 형태를 가지게 된다. main() 함수가 프로그램에 없고 모듈을 커널에 적재하거나 삭제할 때 사용하는 함수가 존재한다.

각 모듈은 대개 하나의 드라이버를 대상으로 만들어지며 여러 함수와 자료 구조로 이루어진 하나의 독립된 프로그램이다. 이러한 독립된 모듈은 설치과정을 통해 커널에 링크 되어 커널에서 실행되는 함수 역할을 수행하게 된다.

저장부분의 디바이스 드라이버는 module 의 등록,

Device open, DATA read, Device close, module 의 삭제로 구성되어 있다. 우선 module 을 등록 하게 되는데 Module 의 등록에서는 드라이버를 커널에 등록하고 I/O 영역이 비어 있는지를 확인 한 후 I/O 영역을 등록하게 된다. 그 다음 Device open 은 open 함수를 호출하여 수행하게 되는데 open 함수는 드라이버가 이후 동작을 준비 할 수 있도록 초기화를 제공한다. Coding 한 디바이스 드라이버의 open 함수에서는 데이터를 읽어오기 위한 순환 큐와 임시 버퍼의 사용을 위한 메모리를 할당하고 사용자 카운트 수를 늘리는 작업을 수행한다. 그리고 추가적으로 PXA250 의 GPIO 를 이용하여 FIFO 를 초기화 시키는 작업도 수행한다. Data read 는 read 함수를 통해서 이루어진다. 우선 HDD 를 마운트 시키고 읽기 영역으로 할당되어 있는 메모리에 있는 데이터를 순환 큐에 넣는 과정을 반복한다. 인터럽트가 걸리면 인터럽트 함수가 수행되는데 인터럽트 함수에서 순환 큐에 저장되어 있는 데이터를 한꺼번에 어플리케이션 공간으로 copy 해 온 후 HDD 가 마운트 된 디렉토리에 file 을 생성하여 저장하게 된다. Device close 함수에서는 인터럽트를 해제하고 사용한 메모리를 반환하는 작업을 수행하게 되며 마지막으로 사용된 module 을 삭제하게 된다.

재생부 디바이스 드라이버는 저장부의 디바이스 드라이버와 거의 유사하며 데이터를 read 하는 대신에 write 를 해주게 되며 데이터를 write 하는 방법도 인터럽트 방식이 아닌 polling 방식을 사용하였다.

IV. 디바이스 드라이버와 시스템 테스트

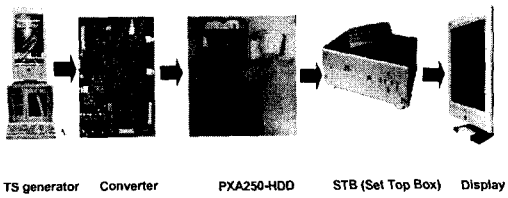

그림 5. 전체 시스템의 구조

HDD 를 이용한 저장·재생기의 전체 시스템의 테스트는 다음과 같은 과정을 거쳐 수행하였다. TS 발생기에서 발생된 트랜스포트 스트림은 Converter 를 거쳐 STB 형태로 변환되고 PXA250 에 의해서 HDD 에 저장 된다. HDD 에 저장된 트랜스포트 스트림은 STB 을 거쳐 디스플레이 된다.

PXA250 에서 데이터를 읽고 쓰는 디바이스 드라이버는 모듈형태로 구성하여 테스트 하였으며 테스트가 끝난 디바이스 드라이버는 ramdisk 이미지 형태로 제작하여 플래쉬에 다운로드 하였다.

V. 결론

본 논문에서는 HDD 를 이용한 저장·재생기를 구현하고 이를 위해 리눅스를 기반으로 한 디바이스 드라이버 프로그램을 작성하였다. PXA250 를 MCU 로 사용하였고 부트로더와 커널과 ramdisk 이미지를 플래쉬 메모리에 탑재하였으며 SDRAM 을 메모리로 사용하였다. IDE 인터페이스를 이용하여 HDD 를 저장 매체로 이용하였으며 추가적으로 PXA250 와 트랜스포트 스트림 간에 발생하는 속도 문제 해결을 위해 FIFO 를 사용하였다. 디바이스 드라이버는 모듈형태로 구성하여 테스트가 용의 하게 하였다.

트랜스포트 스트림을 저장하기 위해 설계한 HDD 를 이용한 저장 재생기는 트랜스포트 스트림이 아닌 다른 데이터를 저장 할 수 있도록 충분히 변형이 가능하며 운영체제를 리눅스를 사용하고 PXA250 를 MCU 로 사용했기 때문에 임베디드 시스템의 기능을 할 수 있도록 구현되어 있어서 다양한 용도의 사용이 가능하다.

참고문헌

[1] http://www.falinux.com.

[2] http://www.kelp.or.kr .

[3] Intel, "Intel PXA255 processor development manual", Intel, 2003.

[4] 박영환, "임베디드 시스템 임베디드 리눅스", 사이텍미디어, 2002.

[5] Alessandro Rubini, "Linux Device Drivers", O'relly, 1998.

[6]Steve Furber, "ARM system-on-chip Architecture 2nd edition", Addison-Wesley, 1994.

[7]Richar& Neill 저/ 이태용 역 "Linux Programming 2nd Edition", 정보문화사, 2000.

[8] Castro Lopo, Aitken and Jones 저/ 박재현·정재원 역 "리눅스 C 프로그래밍", 인포북, 2000.

[9] 박장수, "리눅스 커널 분석 2.4", 가메출판사, 2003.

[10] 권상호, 고성규, 강효성, 민기획, "Unix & Linux C Programming", 영진닷컴, 2003.

[11] http://lxr.lxr.no.

[12] http://kldp.org.