임베디드 리눅스 시스템 상에서의 시리얼 플래시 메모리 파일시스템의 구현

강정주[0] 공기석
한국산업기술대학교 산업기술대학원
guaranaman@mac.com, kskong@kpu.ac.kr

An Implementation of Serial Flash File System in Embedded Linux with MTD Subsystem

Jung Joo Kang[0], Ki-Sok Kong
Graduate School of Industrial Technology, Korea Polytechnic University

요 약

최근 임베디드 시스템이 많은 발전을 하고 있고 그 내부에는 공간적인 장점으로 인하여 플래시 메모리가 많이 사용되고 있다. 시리얼 타입의 플래시 메모리는 공간적인 측면에서 좀더 장점을 보이며 최근에 사용이 늘고 있는 플래시 메모리이다. 오늘날 임베디드 운영체제로 각광 받고 있는 리눅스에는 일반적인 플래시 메모리를 위한 제어기와 파일 시스템을 갖추고 있으나 시리얼 타입의 플래시 메모리의 지원은 이루어지지 않고 있다. 이 논문에서는 임베디드 리눅스에서 기존의 플래시 메모리 프레임 워크를 이용하여 시리얼 타입의 플래시 메모리 파일 시스템을 구현하고 실험을 통하여 시리얼 인터페이스의 성능이 파일 시스템의 성능에 큰 영향을 주는 것을 확인한다.

1. 서론

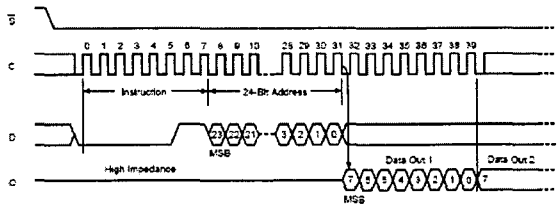
시리얼 플래시 메모리는 어드레스 및 데이터 인터페이스가 단일 시리얼 통신으로 되어 있는 플래시 메모리이다. 처음에는 임베디드 시스템에서 소 용량 시리얼 EEPROM을 대체하는 기능으로 사용되기 시작해서 현재는 대용량의 파일시스템 저장 매체까지 사용 영역을 확대하고 있다. 시리얼 플래시 메모리는 핀 수가 적기 때문에 작은 크기의 칩 생산이 가능하고 병렬(parallel) 버스에 비하여 전기적 잡음을 적게 발생 시키며, 용량에 관계없이 동일한 하드웨어 설계를 할 수 있다는 장점이 있다. 이러한 장점 덕분에 크기, 전기적 잡음에 민감한 응용에 널리 사용되고 있으며, 기술의 발달에 따라 시리얼 플래시 메모리의 사용영역은 더욱 확대 되어 나갈 것으로 예상 된다.

이 논문에서는 리눅스에서 메모리 기반 장치를 사용하기 위한 프레임워크인 MTD(Memory Technology Device)를 이용하여 플래시 메모리용 파일 시스템인 JFFS2 (Journaling Flash File System Version 2)를 사용하기 위한 방법을 제시하며, 실제 구현하여 성능을 평가하고, 또한 더 나은 성능을 위해 고려 되야 할 점을 기술한다. 본 논문의 구성은 다음과 같다. 2장에서는 시리얼 플래시 및 MTD, JFFS에 관한 관련 연구를 살펴보고, 3장에서는 구현을, 4장에서는 시리얼 플래시 메모리를 사용한 JFFS2 파일 시스템의 성능을 측정하고 문제점을 파악하며, 5장에서는 결론을 맺는다.

2. 관련연구

2.1 시리얼 플래시 메모리의 특성

시리얼 플래시 메모리는 동기화된 시리얼 통신을 통신 인터페이스로 사용한다. 시리얼 플래시 메모리에서 가장 널리 사용되고 있는 것은 모토롤러의 SPI(Serial Peripheral Interface)이다. 시리얼 플래시 메모리는 이러한 시리얼 인터페이스로 명령과 어드레스, 데이터를 순차적으로 전송한다. 시리얼 플래시는 CFI(Common Flash Interface)같은 표준이 현재까지는 존재하지 않으며 제조사마다 각기 다른 특징과 명령을 갖지만 명령-어드레스-데이터의 순차 형식은 동일하다. 그림 1은 시리얼 데이터가 어떻게 전송되는 지를 보여준다 [3].



[그림 1] 시리얼 인터페이스를 통한 데이터 전송

2.2 MTD subsystem

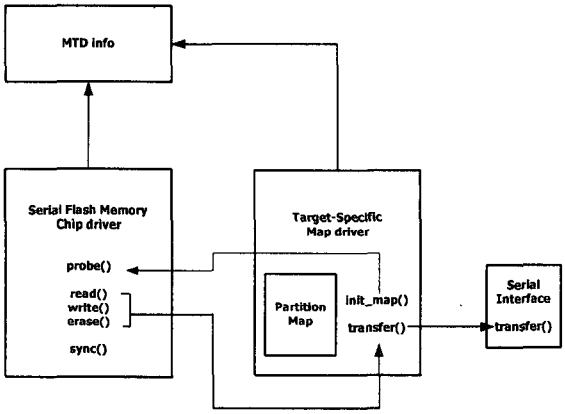
MTD는 리눅스에서 다양한 종류의 메모리 장치를 공통된 인터페이스로 접근하기 위해 설계된 프레임워크이다. MTD는 저 수준의 칩 드라이버와 유저 모듈(user module)이라 불리는 고수준의 인터페이스로 나뉘며 이들은 각각 보다 하위 또는 보다 상위의 계층들과의 인터페이스 역할을 한다 [2]. 칩 드라이버는 다양한 종류의 메모리 장치의 제어 및 데이터 전송을 수행하는 저 수준의 디바이스 드라이버이며 유저 모듈은 칩 드라이버와 보다 상위의 프로그램과의 인터페이스를 담당한다. 유저 모듈은 실제 하드웨어에 해당하는 칩 드라이버를 적재하며, 캐릭터 디바이스, 블록 디바이스, 파일 시스템 등을 포함하고 있다.

2.3 JFFS2

JFFS2는 LFS(Log-Structured File system)의 일종인 Journaling File System을 MTD subsystem에 적용시킨 것이다. 다른 메모리 기반 저장 장치가 파일 시스템을 지원하기 위하여 변환계층을 두는 것과 달리 JFFS2는 LFS를 MTD 장치에 직접 적용시킨다 [2]. 이 밖에 JFFS2는 전체 블록을 고르게 사용하는 wear-leveling, 압축과 Power-fail에 대한 복구 능력을 갖추고 있다 [1].

3. 구현

3.1 전체구조



[그림 2] 시리얼 플래시 파일시스템의 구조

그림 2는 시리얼 플래시 파일시스템의 구현을 위한 메모리 드라이버의 전체 구조를 나타낸다. 시리얼 플래시 메모리 드라이버는 완전히 새로운 개념을 도입하여 작성해야 하는 것은 아니다. 일반적인 플래시 메모리 드라이버가 수행하는 작업을 기본으로 해서 시리얼 플래시 메모리에만 적용되어야 할 부분을 추가 또는 변경하는 것이 효율적이다. 이하의 설명은 기존 플래시 메모리와 차이가 나는 부분을 중심으로 서술한다.

3.2 MTD 칩 드라이버

시리얼 플래시 메모리 칩 드라이버를 위해 일반적인 MTD 칩 드라이버의 가장 기초적인 기능인 probe, read, write, erase, sync를 구현한다. 칩에서 지원하지 않는 suspend, resume과 같은 기능은 배제하였다. 또한 시리얼 플래시 메모리는 현재 표준화된 공통의 인터페이스가 없으므로 칩 별로 독립적인 드라이버를 작성해야 한다.

Probe 기능은 시리얼 플래시 메모리의 초기화 과정을 수행하는데 여기에는 질의를 통한 동적인 어드레싱 방식 세팅을 포함된다. 이 probe는 3.4절에 설명될 맵 드라이버(Map driver)에 의해 호출된다. 맵 드라이버는 시리얼 플래시 메모리 칩 드라이버가 데이터 전송에 사용할 시리얼 데이터 전송 인터페이스를 제공해야 한다. 이 인터페이스는 <명령 송신 - 데이터 수신> 및 <명령 송신 - 데이터 송신>을 기본 형식으로 하는 시리얼 플래시 메모리와 시리얼 인터페이스의 특징을 고려하여 읽기/쓰기를 동시에 수행하는 전 이중 방식의 전송 인터페이스로 결정하였다. read는 읽기 명령과 어드레스를 시리얼 인터페이스로 전송 한 후 연속해서 데이터를 수신하며, write는 쓰기 명령과 어드레스를 전송한 후 연속해서 데이터를 송신한다. 일반적인 플래시 메모리와 달리 시리얼 플래시 메모리의 쓰기는 메모리에서 정의하는 페이지 단위로 이루어진다.

3.3 시리얼 인터페이스

시리얼 인터페이스 드라이버는 CPU에서 제공되는 하드웨어시리얼 인터페이스 드라이버를 구현한다. 시리얼 인터페이스 드라이버는 MTD subsystem의 범위를 벗어나지만 시리얼 플래시 메모리 구현을 위해서는 반드시 필요한 드라이버이다. 본 구현에서는 Intel PXA255 CPU의 내장 하드웨어 시리얼 장치 인 SSP(Synchronous Serial Port)를 사용하였다. 하드웨어의 지원 또는 자원 사용 가능성에 따라 폴링 I/O, 인터럽트 I/O, DMA(Direct Memory Access) I/O방식을 선택하여 사용할 수 있는데, 전체 시스템의 부하를 줄이기 위하여 인터럽트 또는 DMA를 사용 할 수 있으며, 특히 고속의 시리얼 인터페이스를 사용하기 위해서는 DMA를 반드시 사용해야 하는 경우도 있다. 몇 번의 실험을 통하여 얻어진 결론으로는 Intel PXA255 CPU의 경우 2Mbps 이상의 전송 속도를 얻기 위해서는 DMA를 사용하는 것이 바람직하다.

3.4 맵 드라이버

맵 드라이버는 실제 하드웨어의 메모리 정보를 기초로 플래시 메모리 및 파티션을 타겟 시스템에 등록하는 기능을 수행하는 드라이버이다.플래시 메모리의 파티션 정보 또한 맵 드라이버에 명시되는데, 시리얼 플래시 메모리는 Memory Mapped I/O가 아니므로 파티션의 주소공간은 전체 시스템 내에서 유일한 것이 아닌 칩 내에서만 유일하다. 따라서 시리얼 플래시 메모리가 여러 개 있으면 각각의 칩에 대하여 동일한 주소를 할당 할 수 있다. 물론 한 파티션 내에 여러 개의 시리얼 플래시 메모리를 할당하는 방법도 기술적으로는 가능한데 이는 차후에 적용될 예정이다. 또한 맵 드라이버는 칩 드라이버에게 시리얼

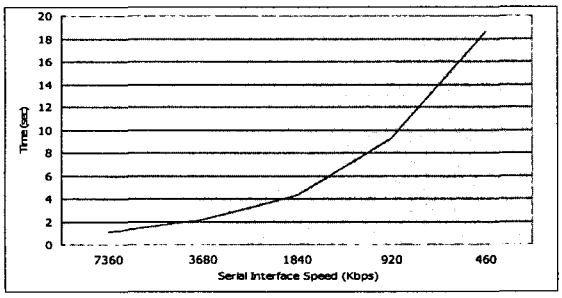
데이터 전송 인터페이스를 제공한다. 시리얼 인터페이스는 타겟 하드웨어에 따라 다양하며, 특정 타겟에 대한 맵 드라이버는 알맞은 시리얼 인터페이스를 칩 드라이버에게 넘겨준다.

4. 실험 및 분석

실험환경은 다음 표 1과 같다.

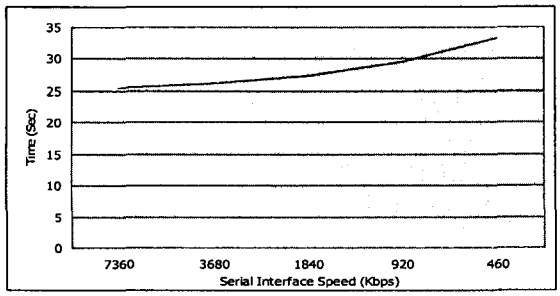
[표 1] 실험환경

Target System	Intel PXA255 400MHz processor, 64MB memory, RAM disk root file system
Linux Kernel	2.4.19
Serial Flash Memory	ATMEL AT45DB321B 32Mbit
Serial Interface	PXA on-chip SSP



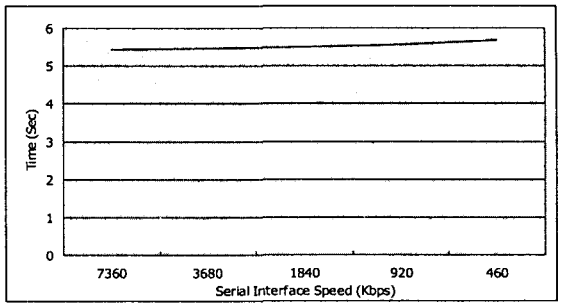
[그림 3] JFFS2 4MB 파티션 마운팅 시간

그림 3은 4MB 크기의 완전히 지워진 파티션을 마운팅하는데 걸리는 시간이다. 마운팅 작업은 전적으로 데이터를 읽는데 걸리는 시간에 좌우되므로 시리얼 인터페이스의 속도가 성능과 비례한다.



[그림 4] 1MByte 데이터 복사 시간

그림 4는 1MByte 크기의 파일 한 개를 시리얼 플래시 메모리 파티션 내로 복사하는데 걸리는 시간이다. 마운트에 비하여 인터페이스 속도의 영향이 상대적으로 적은데, 이는 쓰기에 걸리는 시간이 데이터 전송에 걸리는 시간보다 훨씬 많이 걸리기 때문이다. 시리얼 플래시 메모리는 병렬 플래시 타입처럼 2개의 플래시 메모리를 병렬로 배치하여 버스 크기를 넓혀 2배의 속도를 얻는 방법이 불가능하다. 현재 많은 임베디드 시스템에서 이러한 병렬 배치를 사용한다는 점을 고려하면 이러한 점이 시리얼 플래시 메모리의 약점으로 작용할 수 있다.



[그림 5] 1MB 파일 삭제 시간

그림 5는 1MByte 크기의 파일 한 개를 삭제하는데 걸리는 시간이다. 이 역시 데이터를 전송하는데 걸리는 시간보다 메모리상에 쓰기, 지우기 하는데 걸리는 시간이 크기 때문에 메모리 자체의 성능에 크게 좌우된다.

5. 결론 및 향후 과제

앞에서 살펴본 바와 같이 시리얼 플래시 메모리는 기존의 플래시 메모리와 비교하여 장단점이 분명하다. 시리얼 인터페이스 속도가 읽기 성능을 좌우하고 병렬 연결로 인한 버스 대역 폭 확장의 이점을 얻을 수 없다는 단점이 있지만, 기존의 플래시 메모리 드라이버는 대부분 폴링으로 구현되어 있는 것에 비해 시리얼 플래시 메모리에서는 인터럽트/DMA 방식을 사용할 수 있기 때문에 시스템 부하를 크게 줄일 수다는 것은 큰 장점이다. 또한 임베디드 리눅스 하드웨어 구성시 선택사항을 늘릴 수 있다는 점도 의미가 있다. 현재 시리얼 플래시 메모리에 대한 성능 최적화 작업이 진행 중이고, 향후 여러 개의 시리얼 플래시 칩으로 하나의 파티션을 구성하는 기능을 추가로 개발할 예정이다.

6. 참고 문헌

[1] David Woodhouse, " JFFS: The Journaling Flash File System" , Technical Paper of RedHat Inc., 2001.

[2] Karim Yaghmour, " Building Embedded Linux Systems" , O' Reilly, 2003.

[3] STMicroelectronics, " 8 Mbit Low Voltage, Page-Erasable Serial Flash Memory With Byte-Alterability and a 25MHz SPI Bus Interface" , STMicroelectronics, 2004.