컴팩트 플래시 지원을 위한 Windows CE 부트로더의 설계 및 구현

피무호° 최종필 공기석
한국산업기술대학교 컴퓨터공학과
{pi79°, jpchoi, kskong}@kpu.ac.kr

A Design and Implementation of Windows CE Boot Loader to support Compact Flash

Mu-ho Pi° Jong-pil choi Ki-sok Kong
Dept. of Computer Engineering, Korea Polytechnic University

요 약

Windows CE는 Microsoft사의 Windows 운영체제 가운데서 가장 작은 운영체제로서 일반 데스크톱 Windows 커널을 수용할 수 없는 소형/임베디드 장비에서 주로 사용되어진다. 현재 Windows CE에서 사용되고 있는 부트로더로는 E-boot(Ethernet bootloader)가 있으며 RAM 이미지와 플래시 이미지 다운로드 기능을 제공한다. E-boot의 문제점으로는 플래시 메모리상에서 부팅을 수행하기 때문에 NOR 타입의 플래시만을 지원하며, 컴팩트 플래시와 같은 NAND 타입의 플래시 지원하지 않는다. 이는 OS Binary 이미지의 용량이 NOR 플래시를 초과할 경우에 수행이 불가능하다는 문제를 발생시킨다. 따라서 본 논문에서는 기존의 E-boot를 수정하여 NOR 플래시보다 상대적으로 가격이 저렴하고 휴대성이 좋은 컴팩트 플래시 메모리를 이용하여 부팅이 가능한 부트로더를 구현한다. 또한 컴팩트 플래시 지원을 위한 새로운 읽기/쓰기 메카니즘을 소개한다.

1. 서 론

Windows CE는 Microsoft사의 Windows 운영체제 가운데서 가장 작은 운영체제로서 기존의 Win32 API를 공유하며 소규모 롬 기반으로 동작되는 운영체제이다. 이는 Windows API의 적용 범위를 일반 데스크톱 Windows (Windows Me, Windows NT) 커널을 수용할 수 없는 소형/임베디드 장비 시장으로까지 확장시키게 되었다[1].

Platform마다 하드웨어 구성이 다를 수 있으므로 Windows CE는 초기화 코드의 일부를 OEM(Original Equipment manufacturer)에서 작성하도록 하고 있다. 이 초기화 코드는 커널 아래에 있는 HAL(Hardware Abstraction Layer)에 덧붙여진다. 그리고 이 HAL이 Windows CE 커널 코드에 정적으로 링크되어 커널 이미지가 만들어진다[2].

임베디드 시스템에서의 부트로더는 보드의 초기화와 저장 매체로부터(혹은 시리얼이나 이더넷을 통해서) 커널을 읽어서 메모리의 적절한 위치로 적재한 다음 커널로 제어를 옮기는 역할을 수행한다.

현재 Windows CE에서 사용되고 있는 부트로더로는 E-boot(Ethernet boot)가 있으며 램 이미지와 플래시 이미지 다운로드를 제공하고 있다. E-boot는 플래시 메모리상에서 부팅을 수행하기 위해서 NOR 방식의 플래시 메모리만을 지원한다. 따라서 컴팩트 플래시와 같은 NAND 방식의 플래시 메모리는 사용할 수 없다. 따라서 본 논문에서는 NAND 방식의 플래시 메모리를 사용하기 위한 방법을 제시한다.

컴팩트 플래시를 지원하는 부트로더를 구현하기 위해서는 세 가지 문제가 있다. 첫 번째는 "이더넷을 통해 다운로드되는 패킷들을 어느 시점에 저장할 것인가?", 두 번째는 "이더넷을 통해 다운로드된 데이터들을 어떻게 컴팩트 플래시에 저장할 것인가?", 세 번째는 "컴팩트 플래시에 저장되어있는 이미지를 램 영역으로 어떻게 옮길 것인가?"이다. 본 논문에서는 앞에서 언급한 세 가지 문제들을 해결하고 컴팩트 플래시 지원이 가능한 부트로더를 설계, 구현한다.

본 논문의 구성은 다음과 같다. 2절에서는 관련 연구로 기존 Windows CE binary image 포맷과 E-boot의 부팅과정에 대하여 기술하며, 3절에서는 언급된 세 가지 문제 해결 방안과 부트로더 설계, 구현 방법을 언급하고 4절에서는 구현된 부트로더를 통하여 발생될 기대효과와 향후 과제에 대하여 논한다.

2. 관련 연구

2.1 Binary Image Data Format

Windows CE binary image 파일은 각 섹션들의 데이터 집합으로 이루어져 있으며 각각의 섹션은 시작 주소, 길이, 체크섬으로 이루어져있다. [표 1]은 Windows CE의 binary image 파일의 포맷을 보여주고 있다.

[표 1] Windows CE binary image format

Field	Length(bytes)
Sync bytes (optional)	7
Image header	
Image address	4
Image length	4
Records	
Record address	4
Record length	4
Record checksum	4
Record data	Record length

Sync bytes의 byte 0은 B로 .bin 파일 포맷을 의미하며 1-6 byte들은 0, 0, 0, F, F, ₩n으로 이루어진다. 이미지 헤더는 이미지 주소와 이미지 길이로 이루어지며 이미지 주소는 이미지의 실제 시작 주소를 이미지 길이는 이미지의 실제 길이를 가진다. 마지막으로 레코드는 레코드 주소, 레코드 길이, 레코드 체크섬과 레코드 데이터로 이루어지며 레코드 주소는 데이터 레코드의 실제 시작 주소를 가지며 레코드 데이터는 레코드 데이터의 길이를 레코드 체크섬은 레코드 데이터의 체크섬을 계산하며 마지막으로 레코드 데이터는 실제 레코드 데이터를 의미한다[4].

2.2 E-boot 부팅 과정

본 논문에서 소개할 컴팩트 플래시카드 지원을 위한 Windows CE 부트로더 구현에 앞서 E-boot의 부팅 과정은 다음과 같다. E-boot는 크게 4가지의 제어 흐름으로 이루어진다. 먼저 OEMDebugInit 함수는 부트로더가 시작될 때 처음으로 수행되는 함수로서 serial UARTs와 같은 debug output 하드웨어를 초기화시킨다.

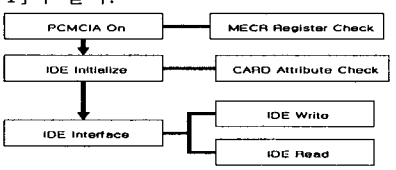
OEMPlatformInit 함수는 clock, driver와 같은 platform에 열거된 디바이스를 초기화하며 OEMPreDownload는 OS 이미지 다운로드를 수행하기 전에 호출되는 함수로서 사용자를 위해 수정될 수 있다. 마지막으로 OEMLaunch 함수는 OS의 시작을 위해 마지막으로 호출되는 함수로서 다운로드된 커널 이미지의 시작 주소로 점프하여 OS를 수행하게 된다[1].

3. 구 현

3.1 부트로더 설계

본 논문에서 소개하는 컴팩트 플래시 지원을 위한 Windows CE 부트로더 구현을 위한 환경은 다음과 같다. 프로세서는 intel PXA255(XScale), 256Mbyte SDRAM, Sandisk사의 512Mbyte 컴팩트 플래시로 하드웨어는 구성되어지며, 개발 툴로는 platform builder .Net 4.2를 사용하였다.

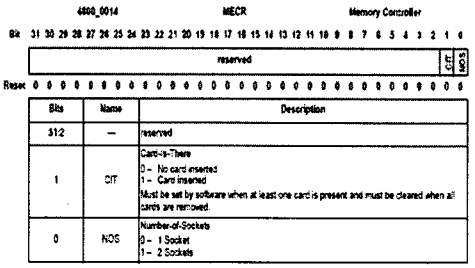
컴팩트 플래시 카드 읽기/쓰기 수행을 위한 제어 흐름은 [그림 1]과 같다.



[그림 1] 컴팩트 플래시 제어 흐름

컴팩트 플래시가 슬롯에 삽입되면 PCMCIA on을 수행시키게 되고 읽기/쓰기를 위하여 IDE를 초기화한다. IDE 초기화가 완료되게 되면 IDE 읽기/쓰기가 가능하게 된다. IDE 읽기/쓰기를 위한 각 제어 흐름의 세부 사항은 다음과 같다. 컴팩트 플래시의 초기화를 위한 설정 레지스터로는 MECR(0x4800_0014) 레지스터를 사용한다.

MECR 레지스터는 확장 메모리(PCMCIA/컴팩트 플래시) 버스 설정 레지스터로서 구조는 [그림 2]와 같다.

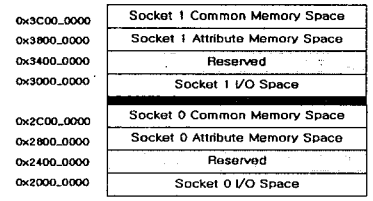


[그림 2] MECR 레지스터 구조

MECR 레지스터는 컴팩트 플래시가 슬롯에 삽입되었을 때, 카드의 삽입여부를 알리는 레지스터이다[3].

메모리 카드의 삽입 상태를 인식하게 되면 IDE 읽기/쓰기를 위한 IDE 초기화 과정을 진행하게 된다.

컴팩트 플래시의 초기화와 읽기/쓰기를 수행하기 위한 Intel PXA255(XScale)의 16bit PC 카드 메모리맵은 [그림 3]과 같다.



[그림 3] PXA255 16bit PC card 메모리 맵

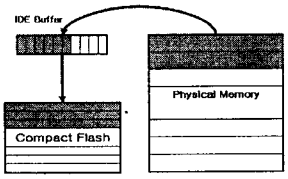
16bit PC 카드 메모리 맵 영역은 8개의 파티션으로 나누어져 있으며 하나의 슬롯은 common 메모리, I/O, attribute 메모리, reserved 공간으로 이루어진 4개의 파티션으로 이루어진다. 각 파티션은 64Mbyte로 제한되어있다[4]. 컴팩트 플래시 카드의 읽기/쓰기 가능 여부는 Attribute 메모리 공간을 통하여 알 수 있게 된다. 만약 읽기/쓰기가 가능하지 않다면 수행을 종료한다. 본 논문에서 사용되어지는 시스템은 한 개의 컴팩트 플래시만을 실험하기 때문에 소켓 0만을 사용한다.

3.2 데이터 쓰기 메카니즘

이더넷을 통해 다운로드되는 이미지 데이터들을 저장하기 위해 기존의 방식인 다운로드 후에 커널 시작 주소로 점프하는 방식이 아닌 확장된 형태의 다운로드 메카니즘을 사용하였다. 이 메카니즘은 컴팩트 플래시에 완전한 형태의 이미지를 쓰기 위해 기존의 이더넷 다운로드가 수행되는 시점에 컴팩트 플래시도 동시에 쓰기를 수행하는 것이다.

하지만 이더넷을 통해 얻어지는 패킷들이 손실되는 경우를 대비하여 체크섬을 통해 검증된 이미지 데이터들만을 컴팩트 플래시에 쓰게 된다. 이 메카니즘을 통하여 본 논문의 첫 번째 문제였던 "이더넷을 통해 다운로드되는 패킷들을 어느 시점에 저장할 것인가?"라는 문제를 해결하였다.

두 번째 문제였던 "이더넷을 통해 다운로드된 이미지 데이터들을 어떻게 컴팩트 플래시에 저장할 것인가?"를 해결하기 위한 방법은 다음과 같다. 다운로드가 수행될 때, 이미지 데이터를 IDE 방식(512바이트 단위로 데이터 쓰기를 수행)으로 쓰기 위한 512바이트의 IDE 버퍼에 저장시킨다. 512 바이트가 채워지면 IDE 버퍼의 내용을 컴팩트 플래시에 쓰게 되며, 다운로드가 완료될 때 까지 이 작업을 반복하게 된다.



[그림 4] C/F write mechanism

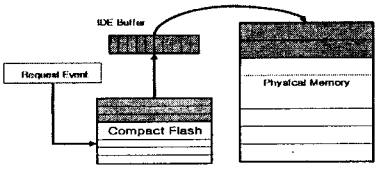
[그림 4]는 데이터 쓰기 과정을 간략하게 보여주고 있다. 다운로드가 이루어지면서 동시에 IDE 버퍼가 채워지며 512바이트 영역을 채우게 되면 IDE 쓰기를 수행하여 실제 컴팩트 플래시 카드에 쓰기를 수행한다.

만약 이미지 다운로드가 완료된 시점에 IDE 버퍼가 채워지지 않았다면 남은 버퍼 영역은 공백으로 처리하며 공백처리 완료 이후에 IDE 쓰기를 수행한다.

3.3 데이터 읽기 메카니즘

세 번째 문제인 "컴팩트 플래시에 저장되어있는 이미지를 램 영역으로 어떻게 옮길 것인가?"의 문제를 해결하기 위한 방법으로는 부트로더에 컴팩트 플래시 다운로드라는 새로운 메카니즘을 추가하여 해결하게 된다.

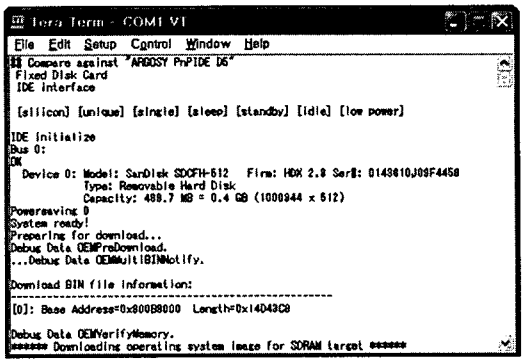
컴팩트 플래시 다운로드 메카니즘은 기본적으로 이더넷을 통해 다운로드 받는 것과 수행 방식은 동일하다. 하지만 IDE 방식에서 읽기는 512바이트 단위로 수행되기 때문에 기존의 이더넷을 통한 다운로드를 그대로 사용할 수가 없다. 따라서 이 문제를 해결하기 위한 방법으로서 컴팩트 플래시 다운로드에서는 IDE 버퍼를 사용하여 이 문제를 해결한다. 이미지 헤더와 레코드와 같은 데이터 다운로드 요청이 오게 되면 IDE 버퍼에 컴팩트 플래시의 저장되었던 내용을 순서대로 블록 단위로 불러오고 데이터의 길이만큼 반복해서 데이터를 IDE 버퍼에 저장시키게 된다. 이와 동시에 IDE 버퍼의 내용은 주소 정보에 따라 실제 메모리에 위치하게 된다. [그림 5]는 데이터 읽기 과정을 간략하게 보여주고 있다.



[그림 5] C/F read mechanism

데이터 다운로드 요청이 오게 되면 컴팩트 플래시 카드에 저장되어 있는 내용을 블록 단위로 IDE 버퍼에 채우게 되며 버퍼가 채워지면 실제 메모리 영역에 올라가게 된다.

3.4 구현 결과



[그림 6] 부트로더 실행 화면

[그림 6]은 실제 부트로더가 수행되어 Windows CE가 부팅되는 과정을 보여주고 있다.

4. 결론 및 향후 과제

본 논문에서는 기존의 Windows CE 부트로더에 컴팩트 플래시 지원을 위한 기능을 추가함에 있어서 생기는 세 가지 문제를 기술하고 이를 해결하기 위한 방법을 제시하였다. 또한 컴팩트 플래시 상에서의 읽기/쓰기를 수행하기 위해 IDE 버퍼를 이용한 확장된 형태의 메카니즘을 제시하였다.

이를 통하여 Windows CE 이미지를 플래시 메모리에 저장할 수 없는 시스템에서도 컴팩트 플래시를 이용하여 Windows CE 시스템을 구축할 수 있다.

현재 구현된 부트로더는 읽기/쓰기 수행 시에 IDE 버퍼를 사용하므로 기존의 부팅 방식보다 약 2배 정도의 수행 시간이 걸린다. 따라서 향후 과제로서 시스템의 부하를 줄일 수 있는 방식의 버퍼 사용 연구를 통하여 시스템의 성능을 향상시킬 예정이다.

참 고 문 헌

[1] James Y.Wilson, "Building powerful platforms with Windows CE", Addison Wesley, 2001.

[2] Douglas Boling, "Programming Microsoft Windows CE 2nd edition", Microsoft Press, 1998.

[3] Intel corporation, "Intel PXA255 Developer's guide"

[4] Microsoft Windows CE Binary image data format, "http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wceosdev5/html/wce50conWindowsCEBinaryImageDataFormatbin.asp"