Windows Embedded CE에서 Remote API를 사용한 Stream Device Driver 동적 로드

*주승화, 전재욱
성균관대학교, 성균관대학교

Stream Device Driver Dynamic Load and Unload by Using Remote API in Windows Embedded CE

SeungHwa Ju, JaeWook Jeon
Sungkyunkwan University, Sungkyunkwan University

Abstract - 본 논문에서는 Windows Embedded CE의 스트림 인터페이스 드라이버(Stream Interface Driver)를 개발하는 새로운 방법을 제안하고자 한다. Windows Embedded CE 디바이스가 부팅되면서 스트림 인터페이스 드라이버는 디바이스 매니저에 의해 메모리로 로드(Load)되고 디바이스가 꺼질 때 메모리로부터 언로드(Unload)된다. 디바이스 드라이버를 변경하여 다시 적용한 후 결과를 확인하고자 할 때 OS 바이너리 이미지를 새로 생성한 후 이미지를 다시 다운로드 하거나 드라이버를 플랫폼 빌더에서 등록한 후 해당 디바이스를 재 부팅하고 Kernel Interface Layer(KITL)를 사용하여 그 결과를 확인할 수 있다. OS 바이너리를 새로 만드는 경우 이미지 생성과 다운로드 시간이 많이 소요되고, KITL를 사용하는 경우 매번 부팅하는 과정을 반복해야 하기 때문에 시간이 더 소요된다. 본 논문에서는 데스크탑에서 Remote API를 이용하고 Stream Device Driver를 디바이스에 전달한 후 그것을 Remote API를 사용하여 로드하고 언로드하는 구조를 제안한다.

1. 서 론

Windows Embedded CE의 주요 구성요소 중 Boot Loader, Kernel, Filesystem, GWES, Device Manager가 주요 커널모드의 구성요소들이다. 이것들을 그림 1과 같이 OEM Adaptation Layer(OAL), Driver들 등이 하드웨어와 연결해주는 역할을 한다[1] [2].

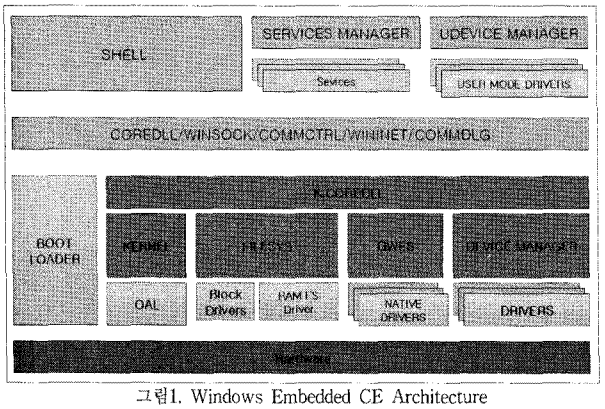

그림1. Windows Embedded CE Architecture

Windows Embedded CE의 Winsock, Coredll등과 같은 유저모드의 Framework Library가 존재하고 그 상위에는 유저모드에서 동작하는 OS Shell, 서비스 매니저(Service.exe)와 유저모드 디바이스 드라이버(UDevice.exe)가 동작하고 서비스 매니저는 서비스들을 관리하고 유저모드 드라이버 매니저는 유저모드에서 동작하는 드라이버들을 관리한다.

하드웨어에 전원이 인가되면 Boot Loader에서 부터 부팅이 시작되고 다음 OAL이 호출된 후 순차적으로 Driver들, 서비스들과 어플리케이션들이 호출된다.

드라이버는 크게 Native Driver와 Stream Interface Driver로 나누어진다. 네이티브 드라이버(Nave Device Driver)는 Graphic Windows Event System(GWES)에 의해 관리되며 OS가 부팅되면서 GWES가 구동된 이후 GWES에 의해 메모리로 로딩되고 디바이스가 켜져있는 동안은 계속 메모리에 적재되어있다. 일반적으로 스트림 인터페이스 드라이버는 디바이스 드라이버 매니저(Device Driver Manager)에 의해 관리되고 부팅을 할때 디바이스드 드라이버 매니저(Device.dll)에 의해 로드되고 디바이스의 파워가 꺼질때 까지 메모리에 적재되어 있다.

디바이스 드라이버가 OS에 적용되기 위해 사용되는 드라이버 중 네이티브 드라이버는 Windows Embedded CE에서 요구하는 고유의 디바이스 드라이버 구조로서 디바이스 드라이버가 제작되어야 하고 네이티브 드라이버는 어떤 역할을 하는 디바이스 드라이버인가에 따라 드라이버구조와 레지스트리 구조는 달라진다.

스트림 인터페이스 드라이버는 공통의 디바이스 구조를 가지고 있다. 다음 그림2 은 스트림 인터페이스 드라이버구조를 나타낸다

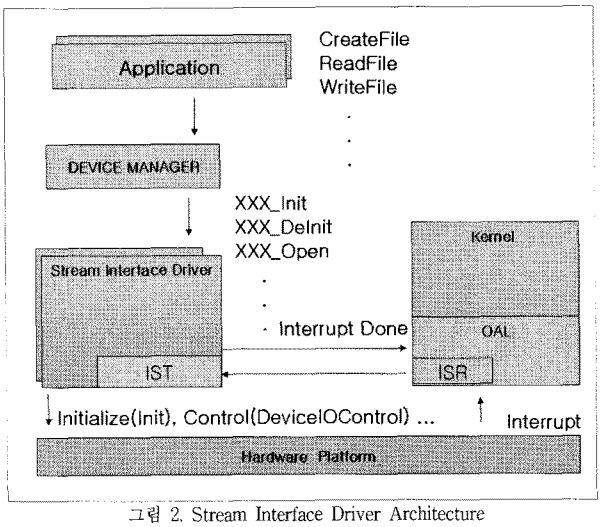

그림 2. Stream Interface Driver Architecture

디바이스 매니저는 XXX_Init을 사용하여 스트림 인터페이스 드라이버를 로딩하려고 시도할 것이다[3]. 그러면 스트림 인터페이스 드라이버 내부의 XXX_Init에서 디바이스 드라이버가 동작하기 위해 필요한 하드웨어 초기화 작업, 메모리 할당, 글로벌변수 선언등의 작업을 수행한다.

스트림 인터페이스 디바이스 드라이버가 더이상 사용되지 않을 경우 메모리에서 언로드 할 수 있다. XXX_Deinit이 호출될때 스트림 인터페이스 드라이버는 메모리에서 언로드 되고 이 과정에서 스트림 인터페이스 디바이스 드라이버가 사용했던 자원들을 해제하고 해당 스트림 인터페이스 드라이버에서 사용했던 주변장치의 파워를 종료시키는 작업등의 하드웨어에 관련된 작업들을 수행한다. 하지만, Windows Embedded CE 디바이스에서 물리적으로 파워가 제거될 경우 XXX_Deinit을 호출할 시간이 없기 때문에 XXX_Deinit이 호출되지 않는다. 그래서 종종 XXX_Deinit에서 리소스를 해제하는 작업을 무시하는 경우도 있다.

본 논문에서는 스트림 인터페이스 드라이버를 동적으로 로드하고 언로드 하고자 하므로 XXX_Deinit에서 리소스를 해제하고 디바이스 주변장치의 파워를 종료시키는 작업을 해주어야 한다.

2. 본 론

디바이스 매니저는 Stream Interface 드라이버가 호출될 때 정의파일(.def)에 정의된 DLLEntry를 호출한 후 XXX_Init을 호출한다. XXX_Init에서는 하드웨어를 초기화하고 메모리와 래지스터 등 디바이스 드라이버가 동작하기 위해 필요한 리소스를 할당하는 역할을 한다.

Windows Embedded CE 디바이스는 Kernel Interface Layer(KITL)와 액티브싱크(ActiveSync)를 사용하여 데스크탑과 통신하기 할 수 있다. KITL은 Windows Embedded CE를 디버깅 하기위해 사용되는 프로토콜이다. 액티브싱크를 사용하기 위해서 디바이스에서는 Ethernet, Serial, Bluetooth, USB들 중 하나의 물리적인 레이어와 디바이스 드라이버가 제공되어야 하고 데스크탑에서도 이와 동일한 물리적 구조와 드라이버가 필요하다. 물리적으로 디바이스와 데스크탑이 연결되면 디바이스는 리모트 어플리케이션(Replog.exe)이 호출되고 데스크탑에서는 액티브싱크 연결관리자(wcecomm.exe)가 호출되어 연결이 설정된다.

KITL 프로토콜이 데스크탑과 디바이스가 연결된 환경에서는 Remote API를 사용할 수 없으며 Remote API을 사용하기 위해서는 액티브싱크가 연결되어 있어야 한다.

그림 3은 RemoteAPI를 사용하여 데스크탑(Desktop)에서 동적으로 새로

운 스트림 인터페이스 디바이스 드라이버를 로딩하는 과정을 보여주고 있다. 우선 액티브 싱크가 연결된 후 데스크탑에서 CeRapiInit으로 초기화하여 RAPI를 사용할 수 있도록 한다[4].

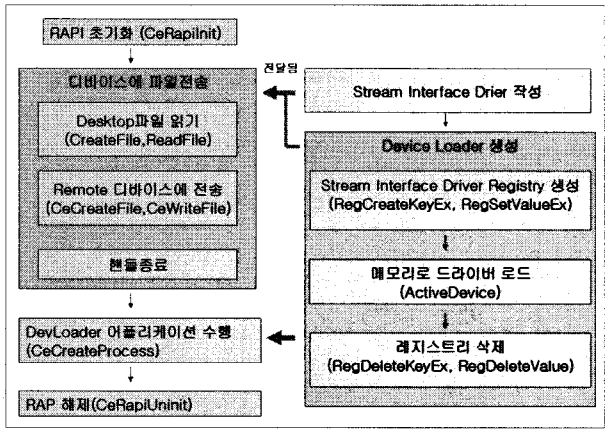

그림 3. RemoteAPI에 의한 Driver 로딩과정

디바이스 드라이버가 동적으로 로드 되도록 하기 위해서 아래 그림 4와 같이 "[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\XXX]" 레지스트리를 생성한다.

```
[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\XXX]
"Prefix"="XXX"
"Dll"="xxxDriver.dll"
"Index"=dword:1
"Order"=dword:0
"Priority256"=dword:95
"SysIntr"=dword:19
"IClass"="{A32943B7-920C-486B-B0E6-92A702A99B35}"
```

그림 4. Stream Interface Driver 레지스트리

여기서 Prefix은 스트림 인터페이스 드라이버를 일반적인 I/O처럼 CreateFile, ReadFile, WriteFile, CloseHandle등을 사용할 때 사용하는 Prefix이다. DLL은 Windows Embedded CE의 시스템 디렉토리인 "\\Windows\"에 저장되는 디바이스 드라이버 명이고 Index은 "XXX1:"처럼 Prefix에서 참조하는 Index순서를 나타낸다. Order는 부팅할 때 여러 디바이스 드라이버들의 로딩 순서를 결정한다.

리모트 디바이스(Remote Device)에서 수행될 디바이스 로더인 어플리케이션(DeviceLoader.exe)은 그림 3에서 보이는 것과 같이 구성되며 다음 처럼 동작한다. RegCreateKeyEx을 사용하여 레지스트리를 만들고 RegSetValueEx을 사용하여 그림 3과 같이 레지스트리 값들을 만든다.

ActiveDeviceEx을 사용하여 스트림 인터페이스 디바이스 드라이버를 메모리로 로드하고 로드 후 반환된 핸들 값을 얻은 후 해당 핸들을 "[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\XXX]" 드라이버의 레지스트리 키 아래 "Handle" 레지스트리 값을 만들어 저장하여 향후 디바이스 드라이버를 언로드 할 때 사용할 수 있도록 한다.

데스크탑에 플랫폼 빌더(Platform Builder)를 통해 미리 생성한 디바이스 로더(DeviceLoader.exe)를 CreateFile, ReadFile을 사용하여 읽은 후 Remote API인 CeReadFile과 CeWriteFile을 사용하여 리모트 디바이스 드라이버에 전달하도록 한다. 그리고 CeCreateProcess을 사용하여 원격으로 DeviceLoader을 수행하고 디바이스 드라이버를 Remote API를 사용하여 데스크탑에서 로드 하도록 한다.

해당 작업이 완료된 후 "...\BuiltIn\XXX]" 레지스트리에 드라이버 로딩정보가 존재하면 부팅 중일때 디바이스 드라이버 메니저는 해당 드라이버를 메모리로 로딩하므로 CeCreateProcess을 수행한 후 우리가 생성한 드라이버의 레지스트리를 "...\BuiltIn\XXX]" 레지스트리에서 삭제한다.

스트림 인터페이스 드라이버가 메모리에 로드된 상태에서는 동일한 드라이버를 다시 메모리로 로드할 수 없다. 그러므로 기존에 미리 저장해둔 핸들을 사용하여 스트림 인터페이스 드라이버를 메모리에서 언로드 해주어야 한다. 스트림 인터페이스 드라이버를 메모리에서 다시 해제하고자 할 경우 다음과 같이 디바이스 메니저와 통신하여 동작하는 언로드 어플리케이션(DeviceUnloader.exe)을 만든다. 그 동작과정은 그림 5와 같다.

기존에 디바이스가 생성될 때 저장해 두었던 디바이스 드라이버의 핸들 값을 RegOpenKeyEx와 RegQueryKeyEx을 사용하여 읽는다. 디바이스의 언로드 함수(DeactivateDevice)를 사용하여 해당 핸들을 메모리에서 해제할 수 있도록 하는 어플리케이션을 만든다. 그리고 Remote API을 사용하여 DeviceLoader.exe를 디바이스에 전달하는 방법과 동일한 방법을 사용하여 DeviceUploader.exe를 디바이스에 전달하다. 그리고 프로세스 생성함수(CeCreateProcess)를 사용하여 메모리에서 스트림 인터페이스 드라이버가 해제되도록 한다.

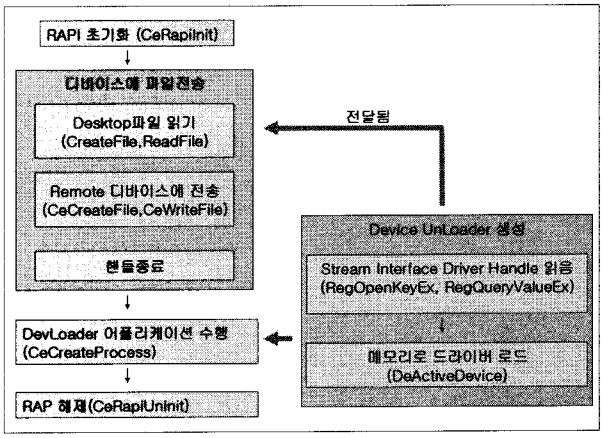

그림 5. RemoteAPI에 의한 Driver 언로딩과정

3. 결 론

Windows Embedded CE에서 사용하는 대부분의 디바이스 드라이버는 Stream Interface Driver이다. 이것을 개발하기 위해서 종종 Kernel Interface Layer 프로토콜(KITL)을 사용한다. 이 프로토콜을 사용하는 경우 OS는 더욱 무거워지고 성능은 감소한다. 특히 OS의 성능에 영향을 받는 카메라 디바이스 드라이버, 오디오 디바이스 드라이버등을 작성하고 디버깅해야 하는 경우 KITL를 사용하지 않고 디바이스 드라이버를 개발해야 하는 경우도 발생한다. KITL를 사용하지 않을 경우 일반적으로 디바이스 드라이버를 변경하고 난 후 변경사항을 확인하기 위해서는 OS를 다시 만들어야 한다. OS를 새로 만드는 시간은 환경에 따라 차이가 있지만 많은 시간이 소요된다.

이미지가 생성된 후에는 그것을 다시 디바이스에 다운로드하여야 한다. OS 바이너리를 다운로드 하는 시간도 시간이 소요되는 일이다.

다른 방법으로 레지스트리를 수정하여 적용하는 방법이 있다. 하지만 이것은 Remote Registry Editor에 접속하여 레지스트리를 변경해 주어야 하고 재 부팅하는 과정을 거쳐야 하므로 시간이 많이 걸리고 번거로운 작업이다.

일반적으로 사용하는 이미지 재 생성과 레지스트리 수정의 방법 대신 이 논문에서 제안한 방식을 사용한다면, 이미지를 다시 만들고 다운로드하거나 디바이스를 재 부팅하는 것 없이 데스크탑에서 동적으로 스트림 인터페이스 드라이버를 로딩과 언로드 할 수 있다. 이 방식은 특히 OS 사이즈가 큰 경우와 부팅시간이 많이 걸리는 경우 유용하다. 또한 디맨드 페이징(Demand Paging)의 OS구조를 사용하지 않아 RAM으로 OS이미지를 로딩하는 경우와 KITL를 사용할 수 없는 경우등 많은 경우에 효과적으로 사용될 수 있다. 향후 우리는 네이티브 디바이스 드라이버(Native Device Driver)의 구조를 분석한 후 네이티브 디바이스 드라이버도 동적으로 로드와 언로드를 한다면 더욱 효과적일 것이다. 이것은 향후 연구과제로 남겨두기로 한다.

[참 고 문 헌]

[1] Alassaad, Z.M. Saghir, M.A.R "CERPID: a reconfigurable platform interface driver for Windows CE. NET", Information and Communications Technology, 2005. Enabling Technologies for the New Knowledge Society: ITI 3rd International C onference, On page(s): 839- 850, Publication Date: 5-6 Dec. 2005

[2] http://msdn.microsoft.com/en-us/library/ms905093.aspx

[3] Lim, Samsu; Chon, Sunghwan; Choi, Hyunsok; Ham, Woonchul, "A development of power button device driver based on Windows CE device driver", ICMIT 2005: Information Systems and Signal Processi ng. Edited by Wei, Yunlong; Chong, Kil To; Takahashi, Takayuki. Pr oceedings of the SPIE, Volume 6041, pp. 142-147, 12/2005

[4] http://msdn.microsoft.com/en-us/library/aa458022.aspx