Sources 파일에 대해서

드라이버 파일을 컴파일 하기 위해서 사용되는 필수 파일들 중 하나가 Sources라 불리는 파일이다. 이 파일은 드라이버를 컴파일 하기 위해 반드시 필요한 파일로써, 드라이버를 컴파일 하기 위한 BUILD 유틸리티가 사용하기 위한 기본 정보들이 포함되어 있다.
Sources 파일은 드라이버를 컴파일 하기 위한 명령어인 "build" 파일에서 인식할 수 있는 매크로 형식으로 구성 되어 있으며, 기본 포맷은 다음과 같다.

```
MACRONAME = MacroValue
```

여기서 사용하는 MacroValue는 텍스트 문자열로써 BUILD 유틸리티가 내부 스크립트에서 MACRONAME을 대체하기 위해 사용된다. 그리고 한 라인 이상의 MacroValue 문자열을 정의하기 위해서는 각 라인의 마지막에 백슬러쉬(₩) 문자를 넣으면 된다.

Sources 파일 내부에서 사용되는 각 매크로 이름에 대해서 살펴보자.

1. TARGETNAME

   TARGETNAME은 파일 확장자를 포함한 컴파일 되는 라이브러리 이름을 정의하며, 반드시 정의되어야 하는 매크로이다. 드라이버 초보 강좌의 첫번째 문서인 "Device Driver 컴파일 환경 설정"에서 설명한 내용 중 세 번째 방법, 즉 드라이버를 컴파일 하기 위해 비쥬얼 스튜디오에서 MakeFile 형식으로 프로젝트를 생성 했다면, 이때 설정하는 MakeFile 프로젝트의 컴파일 옵션 중에서 생성될 드라이버 이름을 지정하는 부분이 있다. 그러나 이때 설정한 드라이버 이름은 드라이버 컴파일 시 적용되지 않고, TARGETNAME에서 설정한 파일 이름으로 바이너리 파일을 생성한다.

2. TARGETPATH

   생성될 모든 파일들이 저장될 폴더 이름을 정의한다. 그러나 BUILD 유틸리티는 대부분 여기서 지정한 폴더에 플랫폼에 특성화된 서브 폴더를 생성한다. 즉, Sources 파일이 포함된 폴더의 하부 폴더로 ₩obj 폴더를 생성한다. 이 매크로 역시 Sources 파일에서 반드시 정의되어야 하는 매크로이다.

3. TARGETPATHLIB

   프로젝트 빌드 시 다른 프로젝트에서 참조하기 위하여 생성된 라이브러리가 저장될 폴더와 이름을 지정한다. 이 매크로가 정의되지 않으면 드라이버에서 사용할 라이브러리는 다른 파일처럼 동일한 서브폴더(TARGETPATH에서 지정한 폴더의 서브 폴더)에 저

장된다.

4. TARGETTYPE

빌드될 프로젝트의 종료를 정의한다. 이 매크로는 일반적으로 LIBRARY나 DYNLINK를 지정하지만 다음과 같은 값을 설정할 수 있다. 이 값 역시 반드시 Sources 파일에서 사용해야 하는 매크로이다.

값	내용	파일확장자
PROGRAM	User-mode 프로그램	.exe
PROGLIB	다른 프로그램에서도 사용할 수 있도록 함수들을 정의한 실행 가능한 프로그램	.exe
DYNLINK	동적 링크 라이브러리(DLL)	.dll
LIBRARY	다른 코드와 연결될 코드를 사용하기 위한 라이브러리	.lib
DRIVER_LIBRARY	드라이버에서 사용하기 위한 라이브러리. LIBRARY와 DRIVER_LIBRARY의 차이점은 포함된 헤더 파일을 위한 기본 위치이다.	.lib
DRIVER	Kernel-mode 드라이버	.sys
EXPORT_DRIVER	다른 드라이버에서 사용할 수 있도록 함수를 정의한 Kernel-mode 드라이버	.sys
HAL	Hardware Abstraction Layer	.dll
BOOTPGM	Kernel-mode 드라이버	.exe
MINIPORT	Ntoskrnl.lib나 hal.lib를 참조하지 않는 Kernel-mode 드라이버	.sys
GDI_DRIVER	Win32k.sys를 사용하는 Kernel0mode 그래픽 드라이버	.dll

5. TARGETEXT

*.cpl*과 같이 DLL을 위한 파일 확장자 이름을 정의한다. 이 매크로가 정의 되지 않으면 DLL을 위한 기본 확장자 이름은 *.dll*이다.

6. TARGETLIBS

프로젝트에서 반드시 참조해야 되는 라이브러리를 정의한다. 이 매크로의 사용은 다음과 같다.

```
TARGETLIBS=$(SDK_LIB_PATH)₩kernel32.lib ₩
```

```
$(SDK_LIB_PATH)₩advapi32.lib ₩
$(SDK_LIB_PATH)₩user32.lib    ₩
$(SDK_LIB_PATH)₩spoolss.lib
```

(2000 DDK 문서에는 이 매크로가 Sources 파일에서 반드시 필요한 매크로라고 정의되어 있다. 그러나 TARGETTYPE가 DRIVER로 설정되어 있는 경우, 드라이버에서 참조할 라이브러리가 없다면 사용하지 않아도 컴파일 하는데 이상은 없다.)

7. INCLUDES

컴파일 하는 동안 참조할 헤더 파일의 위치를 정의한다. 여기서 정의된 목록은 세미콜론(;)으로 분리되며, 지정된 위치는 헤더 파일이 위치한 전체 패스가 될 수 있고(절대적인 위치) Sources 파일이 있는 폴더를 기준으로 하여 정의(상대적인 위치)할 수 있다.

8. SOURCES

컴파일 하고자 하는 파일을 지정하는 매크로로, 반드시 Sources에서 사용되어야 하는 매크로이다. 이 소스 파일은 생성될 DLL이나 라이브러리를 구성한다. 각각의 소스 파일은 빈 영역이나 탭으로 구분되고, Sources 파일이 위치한 폴더에 있어야만 한다.

9. UMTYPE

생성될 프로젝트의 종류를 설정하며, 사용되는 값은 다음과 같다.

windows	User-mode에서 동작하는 Win32 프로그램을 나타낸다.
NT	Kernel-mode에서 동작하는 프로그램을 나타낸다.
console	Win32 콘솔 어플리케이션을 나타낸다.

10. UMAPPL

메인 함수에 포함될 소스 파일의 목록을 포함한다. 이런 파일 이름은 파일 확장자 없이 정의되고 아스테릭 문자(*)에 의해 분리된다. 이러한 목록에 있는 각각의 파일은 SOURCES 라인이 생성한 DLL이나 라이브러리에 링크되고 컴파일 된다. 만약 이 매크로를 사용한다면, BUILD는 자동으로 실행 가능한 파일을 생성한다. BUILD 명령 라인에 실행 가능한 파일 이름을 정의하는 것은 필요하지 않는다.

11. UMTEST

메인 함수에 포함될 소스 파일의 목록을 포함한다. 이런 파일 이름은 파일 확장자 없이 정의되고 아스테릭 문자(*)에 의해 분리된다. 이러한 목록에 있는 각각의 파일은 SOURCES 라인이 생성한 DLL이나 라이브러리에 링크되고 컴파일 된다. 만약 이 매크

로를 사용한다면, BUILD 명령 라인에 파일 이름을 정의함으로써, 생성하고자 하는 파일 이름을 확인해야만 한다.

12. UMAPPLEXT

*.com*이나 *.scr*처럼 실행 파일을 위한 파일 확장자 이름을 정의한다. 이 매크로가 정의되지 않으면 실행 가능한 파일의 확장자 이름은 *.exe*이다.

13. UMLIBS

UMTEST나 UMAPPL에서 정의한 파일과 연결하기 위한 라이브러리의 경로 이름의 목록을 포함한다. SOURCES 파일이 정의한 라이브러리는 이곳에 포함되어야 한다. 이 목록에 있는 각각의 라이브러리는 빈 영역이나 탭에 의해 분리된다. 여기서 정의된 전체 경로는 절대적인 경로이다.

14. NTPROFILEINPUT

링커가 이미지 파일에 저장된 기능들을 어떤 순서로 열거할지에 대해서 파일 목록을 지정할 수 있도록 한다. 이 매크로는 다음과 같이 사용한다.

```
NTPROFILEINPUT=1
```

이 매크로가 정의되어 있다면, Sources 파일에 포함된 폴더는 TargetName.prf라 불리는 파일을 포함해야 한다. 여기서 TargetName은 TARGETNAME에서 설정된 이름이다.

15. DLLORDER

링커가 이미지 파일에 저장된 기능들을 어떤 순서로 열거할지에 대해서 파일 목록을 지정할 수 있도록 한다. 이 매크로는 순서 리스트에 포함된 파일의 이름을 설정해야만 한다. 이 매크로는 NTPROFILEINPUT 매크로 대신 사용할 수 있다.

16. 386_WARNING_LEVEL

컴파일러의 warning 레벨을 설정한다. 기본 값은 다음과 같다.

```
386_WARNING_LEVEL=-W3
```

UMLIBS 경로명은 특별한 방법으로 선언되어야 한다. 왜냐하면 BUILD는 여러 종류의 하드웨어 플랫폼에 맞게 프로젝트 결과물을 생성할 수 있기 때문에, 생성될 결과물에 대한 저장 위치를 다음과 같이 구성된다.

```
%TARGETPATH%₩<cpu_type>₩
```

여기서 TARGETPATH는 Sources 파일에 정의되어 있고, cpu_type은 프로젝트가 생성된 플랫폼을 나타낸다. 예를 들어, TARGETPATH가 LIB로 선언되고, 빌드 요청이 i386을 위한 것이라면, 빌드된 프로젝트는 다음과 같은 서브 폴더를 가리킬 것이다.

```
Lib₩i386₩
```

이러한 약속 때문에, UMLIBS 엔트리는 다음과 같은 형식으로 라이브러리 이름을 나타내야만 한다.

```
<targetpath>₩*₩<library-name>
```

targetpath는 Sources 파일에 있는 TARGETPATH에 할당된 값으로 인식되고, library_name은 실행파일에 링크된 라이브러리의 전체 파일 이름이다. BUILD는 "*"을 프로젝트가 생성될 플랫폼 종류로 대체한다.
특별한 일련의 빌드 프로젝트를 생성하기 위해서는, Dir 파일이나 Sources 파일을 포함하는 폴더로 변경하고 BUILD를 실행시키면 된다. BUILD는 각 서브 폴더의 Sources 파일에 정의된 프로젝트를 자동으로 생성한다.
예를 들어, Sources 파일이 "mylib"라 불리는 라이브러리를 생성할 것을 가리킨다면, 다음과 같은 명령어를 사용하여 라이브러리를 생성할 수 있다.

```
build -386
```

만약 Sources 파일이 "myexe"라는 실행파일을 생성하라는 UMAPPL 매크로를 포함한다면, 위와 같은 명령어는 "myexe" 파일을 같이 생성한다. 이와 반대로, "myexe" 파일을 생성하라는 내용이 UMAPPL 매크로 대신 UMTEST 매크로로 정의되어 있으면, 다음 명령과 같이 빌드 동작에 "myexe"를 포함해야 한다.

```
Build -386 myexe
```

하나의 소스 파일로 구성된 간단한 프로그램을 위해서는 라이브러리 생성이 요구되지 않는다. 이런 환경에서는 다음과 같이 Sources 파일을 정의한다.

1. TARGETNAME과 TARGETPATH를 라이브러리를 생성했던 것처럼 설정한다.
2. TARGETTYPE을 UMAPPL_NOLIB로 설정한다.

3. SOURCES에는 아무것도 설정하지 않는다.
4. UMLIBS에 TARGETNAME 라이브러리를 포함하지 않는다.
5. UMAPPL이나 UMTEST를 사용하여 소스 파일 목록을 작성한다.
6. BUILD를 호출하는데, UMTEST 매크로를 사용했다면 파일 이름을 명시한다.

위와 같이 하면 BUILD는 실행 파일을 생성할 것이다.

Sources 파일은 디바이스 드라이버 뿐만 아니라 다른 종류의 프로그램(*dll, exe, scr* 등등)들도 컴파일 할 수 있도록 구성되었기 때문에, 생성하고자 하는 프로그램에 따라 적절한 매크로를 사용해야 한다.
일반적으로 디바이스 드라이버를 만들기 위해 Sources 파일에서 기본적으로 사용하는 매크로는 다음과 같다(여기서 사용하는 Sources 파일의 내용은 첫 번째 드라이버 강의 "Device Driver의 컴파일 환경 설정"에서 사용했던 샘플 소스의 내용이다).

```
TARGETNAME=HTSys
TARGETPATH=obj
TARGETTYPE=DRIVER

INCLUDES=$(BASEDIR)₩inc;$(BASEDIR)₩inc₩ddk;..₩

SOURCES=HTSYS.C
```

여기서 사용하는 INCLUDES 매크로는 사용하지 않아도 무방하지만, 대부분 디바이스 드라이버에서 참조하는 헤더 파일은 DDK가 설치된 폴더에 있기 때문에 DDK의 헤더 파일이 있는 위치를 선언하였다.
다음에는 일반적인 DDK 문서에 나와 있지 않지만, 디바이스 드라이버 개발 시 사용되는 유용한 팁과 매크로에 대해서 설명하겠다.