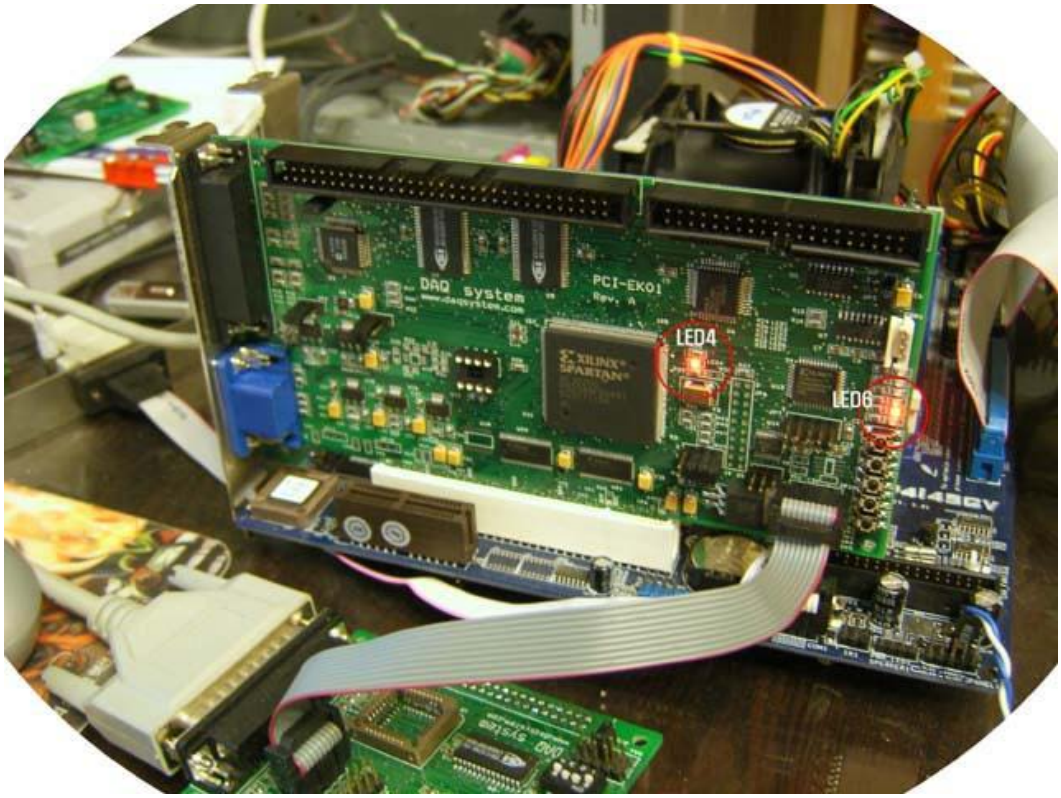
How to build application using API



Benefits of using client DLL for building application

DAQ 보드를 이용한 사용자 프로그램을 작성시 직접 보드의 레지스터를 제어하여 프로그램을 할 수 있으며 이를 "레지스터 레벨 프로그램밍" 방법이라 한다. 다른 방법으로는 보드 제작회사에서 제공하는 클라이언트 DLL에서 제공하는 API(Application Programming Interface)s 를 이용하여 프로그램 하는 방법이 있으며 이를 "API 프로그램밍" 이라 한다.

"레지스터 레벨 프로그램밍" 의 이점은 보드의 기능을 세세히 제어할 수 가 있다. 하지만 보드의 각 레지스터를 자세히 숙지하여야 하므로 프로그램을 하는데 시간이 많이 소요되는 단점이 있고, 프로그램 코딩 시 실수 할 확률이 높아진다. 또한 프로그램을 잘못 작성 시에는 시스템다운 같은 치명적인 문제를 야기할 수 있다.

"API 프로그램밍"의 장점은 보드의 각 레지스터 기능을 숙지할 필요 없이 빠르고 쉽게 사용자 프로그램을 만들 수 있다. 단점으로는 세밀하게 기능을 제어할 수 없으며, 문제가 발생시 자체적인 디버깅이 어려울 수 있다. 이러한 방식을 프로그래머가 주로 선호 한다.

보드 제작 회사에서 제공하는 클라이언트 DLL(API)은 무상으로 제공하는 것도 있지만 각 업체에 따라서는 구입을 하여 사용하여야 하는 경우도 있다. 디에이큐 시스템에서는 기본적으로 클라이언트 DLL을 무상으로 공급을 하고 있으며, 레지스터 레벨의 프로그램밍을 돕기 위한 매뉴얼도 제공하고 있다.

How to link the client DLL to the program project

클라이언트 DLL을 사용자 프로그램에 링크시키기 위하여는 두 가지 방법이 있다. 첫 번째는 암시적 연결 이며, 두 번째는 명시적 연결 방법이다. 향후 설명은 Windows 운영체계 와 Visual C++를 기준으로 설명을 한다.

1. 암시적 연결 방법

(1) 먼저 기본 프로그램의 프로젝트의 뼈대를 만든다.

(2) 프로젝트에 디에이큐 시스템에서 제공하는 다음의 파일들을 "xxxx.dll", "xxxx.lib", "xxxx.h" 를 복사한다.(xxxx는 각 보드의 명칭을 주로 사용한다.)

xxxx.dll : 클라이언트 동적 링크 라이브러리

xxxx.lib : 임포트 라이브러리

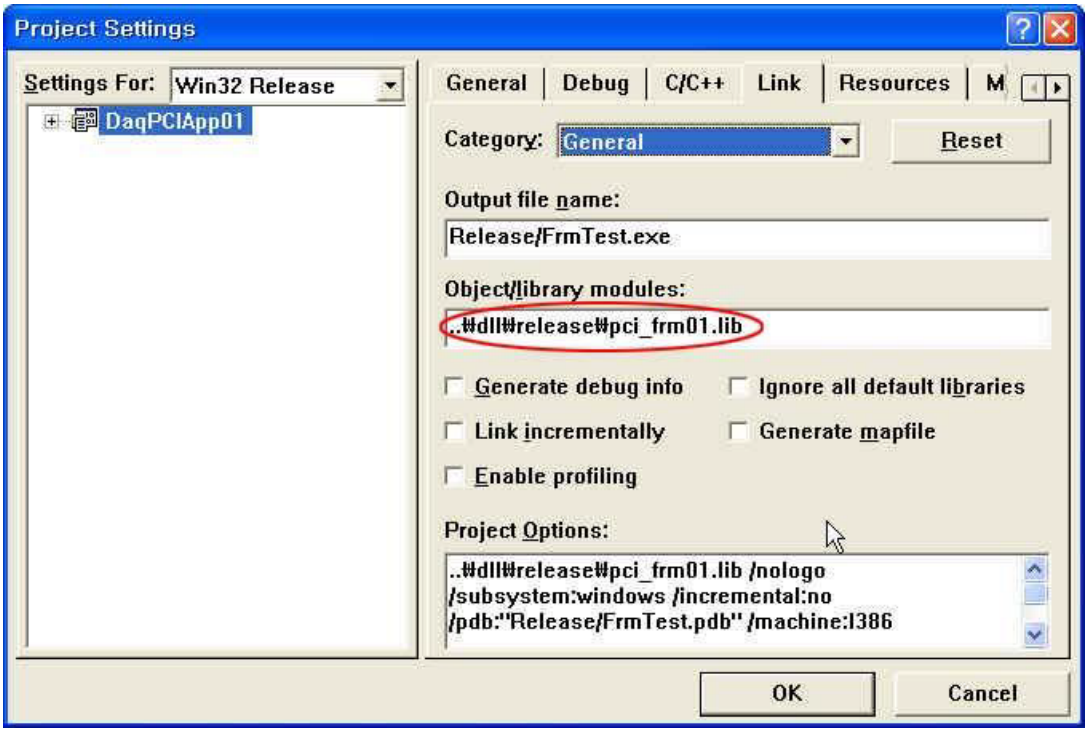
xxxx.h : 함수 정의 헤더

(3) 각 파일을 이용하는 방법은 다음과 같다.

동적 링크 라이브러리는 실행 파일의 폴더나 시스템 폴더에 복사하여 프로그램이 실행될 때 링크 될 수 있도록 한다. 시스템에서 "DLL"을 찾는 순서는 다음과 같다.

<1> 사용자 프로그램이 있는 폴더
<2> 프로세스의 현재 폴더
<3> 윈도우 시스템 폴더
<4> PATH 환경변수에 지정된 폴더

임포트 라이브러리는 프로젝트 세팅에서 설정하여 프로그램 링크할 때 사용되도록 한다. 아래그림 에서 붉은 색 원안의 내용과 같이 설정한다. 그림에서는 “pci_frm01.lib”가 사용되었지만, 각 보드에 맞는 라이브러리 파일 이름을 설정한다.



마지막으로 임포트 헤더 파일을 API 을 호출 하는 각 프로그램 소스에 아래와 같이 “#include” 문을 이용하여 포함 시킨다.

```
#include "frm01_import.h"
```

㈜ 임포트 라이브러리에는 정적 라이브러리와는 달리 실제 실행 코드 이미지가 있는 것이 아니라 동적 링크할 때 필요한 함수 정의에 대한 내용을 담고 있다.

2. 명시적 연결 방법

암시적 방법에는 프로젝트 셋팅에서 DLL파일의 이름이 설정이 되어 있어서 손쉽게 필요한 DLL을 연결할 수 있다. 명시적 연결방법은 프로젝트 셋팅을 하지 않고 DLL파일 이름을 프로그램 소스에서 지정한다.

(1) DLL을 링크 시키기 위하여는 아래와 같이 Windows API인 "LoadLibrary"함수를 사용한다.

```
Handle = LoadLibrary("pci_frm01.dll");
```

LoadLibrary함수는 DLL 파일이 연결되면, 핸들 값을 리턴하며, 실패할 경우에는 NULL값을 리턴한다.

(2) DLL사용이 끝나, 링크를 해제하기 위하여는 아래와 같이 Windows API인 "FreeLibrary"함수를 사용한다 인수는 LoadLibrary에서 획득한 핸들을 이용한다.

```
FreeLibrary(Handle);
```

(3) DLL 내에 있는 함수를 사용하려면 GetProcAddress를 이용하여 함수의 포인터를 얻는다. 인수는 LoadLibrary에서 획득한 핸들을 이용한다.

```
DWORD count;
unsigned char buffer[0x600000];
BOOL returnVal;
typedef BOOL (myfunction)(DWORD *nCnt, unsigned char* buf);
myfunction* pfunc;
pfunc = (myfunction*)GetProcAddress(Handle, "LVDS_GetFrame");

returnVal = (*pfunc)(count, buffer);
```

㈜ 명시적 연결보다 암시적 연결방법이 사용하기 쉬우므로 선호되는 방법이다. 하지만 경우에 따라서는 프로그램에서 동적 연결 라이브러리의 링크/해제를 수시로 하여야 할 경우에는 명시적 연결 방법을 사용한다.