说明：该接口函数全部封装在PCI\_E.h和PCI\_E.cpp文件里。其中界面用到的函数接口如下：

在需要的模块里先定义一个PCIE类。如下

pcie pciee; //然后用该类实例去交换，该实例有下面的这些函数

1：bool **OpenDevice(DWORD vendorid=0x10ee,DWORD deviceid=7)；**该函数用于打开PCIE设备，参数1是硬件的厂商号，参数2是硬件设备号。

该函数执行如果成功，返回TRUE，如果返回FALSE，则调用该函数的程序必须返回。

2：分配内存空间，这里提供了两个函数

当需要DMA读内存的时候（数据从PC机内存流向PCIE），调用下面的函数；

BOOL DMAReadMenAlloc(UINT menBlocknum=4,UINT blocksize=32\*1024)；

说明：该函数执行成功则返回TRUE，若失败调用该函数的程序必须返回；

当需要DMA写内存时（数据从PCIE流向PC机内存），调用下列函数；

BOOL DMAWriteMenAlloc(UINT menBlocknum=6,UINT blocksize=32\*1024)；

说明：该函数和上面DMAReadMenAlloc使用条件一样；

3：设置一次传输过程数据的传输量。调用下列函数；

Void SetTransBlkNum(UINT num);

说明：该函数无返回值，参数的单位值与2设置缓冲区两个函数之一的第二个参数有联系。如果2步的第二个参数是32K，本函数的参数为1024，则表示一次传输过程将要传输32K×1024=32MB数据量，

4：启动PCIE开始传输数据，调用

BOOL StartDMA(BOOL wmen,HANDLE hf=NULL,BOOL isrealop=TRUE,HANDLE finishEven=NULL)；

说明：该函数调用会立即返回。如果成功则返回TRUE，失败则返回FALSE，

参数1：是写内存还是读内存，该参数因该与2部分的函数对应，若2调用的是DMAReadMenAlloc，则该参数必须传FALSE，若2部分调用的是DMAWriteMenAlloc，则必须是TRUE。

参数2：与数据传输有关的文件句柄，若调用DMAWriteMenAlloc，且你需要将从PCIE传到内存的数据保存，则必须在调用4部分函数前创建一个新文件句柄并将该句柄传赋值给第二个参数，则启动后DMA将向该文件写入3部分指定大小的文件数据。若你不需要该文件，可以传入NULL指针；注意：当你需要将本地磁盘上的数据传输到PCIE时，则第二个参数必须是一个有效的文件句柄，当调用4函数后，PCIE将从该文件中读取3函数设定的指定块数的数据。

参数3：是实际工作方式还是模拟工作方式：

如果为TRUE，则数据将通过与PCIE卡上的光纤发送与接受，如果为模拟，则数据在PCIE内部完成操作（无论写还是读）

参数4：事件句柄，如果你调用完4函数，你需要等待DMA完成任务后再继续后续工作，可你又不想做一个while循环等待，则你创建一个事件，将事件句柄传入该参数，然后等待该事件即可。该参数可以设置为NULL。

上面四个函数就可以启动PCIE。下面几个函数需要说明一下；

5：查询当前已完成的块数，

UINT GetTransBlock()；返回当前已传输的数据块数。每块的大小与2函数其中之一的第二个参数相同值

6：查询当前已用时间，

UINT GetTimes() 返回从启动DMA到当前已过去的时间，单位是秒。

7：关闭DMA，void pcie::StopDMA() 该函数只在正常情况下需要中途终止DMA的时候调用。其结果是迫使当前数据传输被任务放弃，注意使用