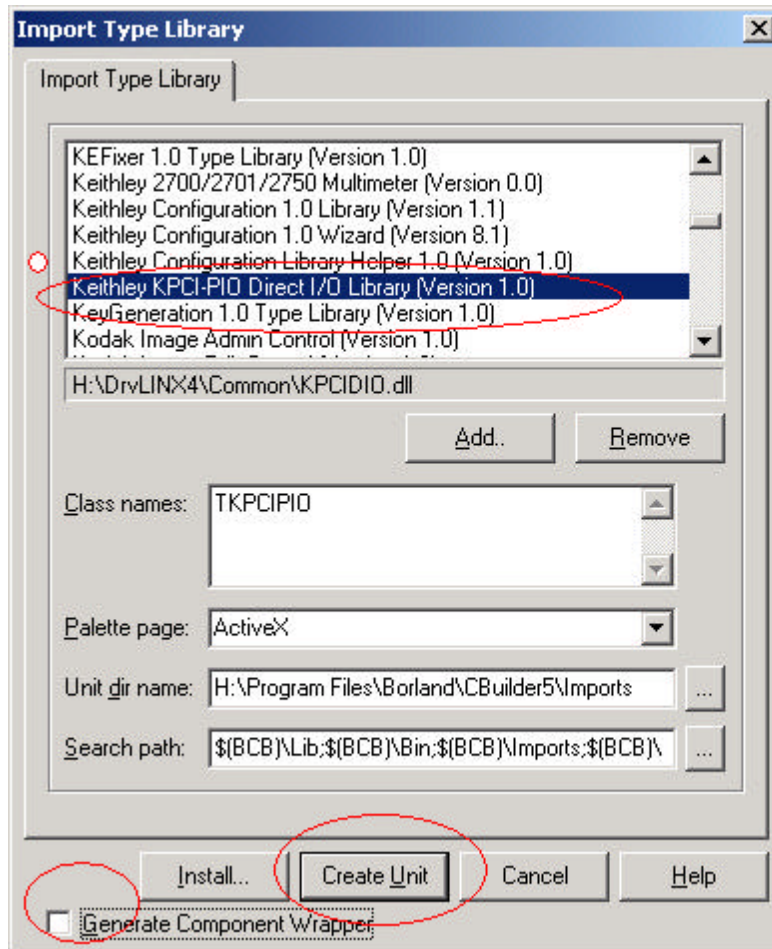


Using Direct I/O COM API of DriverLINX in Borland Builder

Components Used:

Windows 2000 SP1
DriverLINX version: KPCIPIO-850A04
Borland Builder 5.0 Standard
KPCIPIO24

First step is to use the Import Type Library feature on the Project Menu within the Borland IDE. Below is a screen capture:



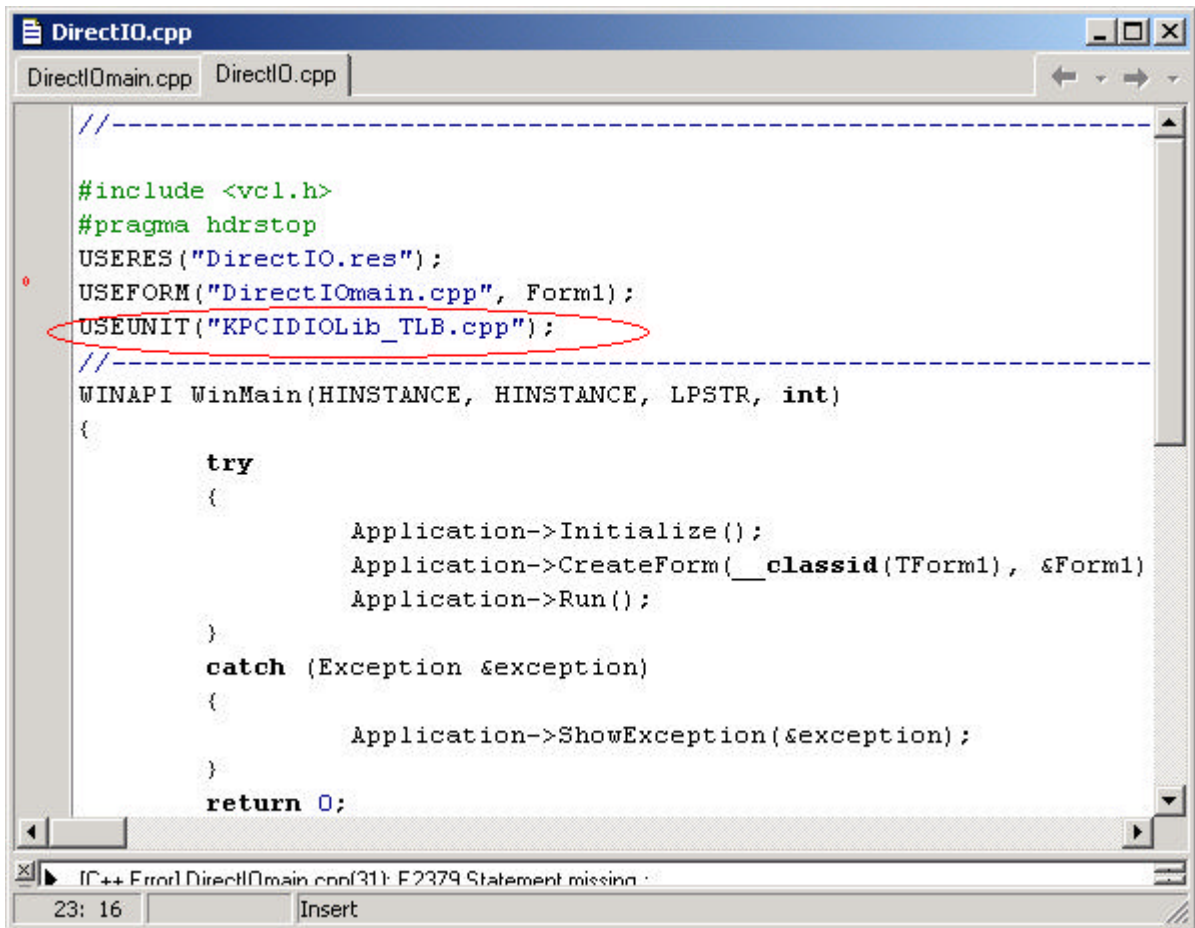
Scroll down to and select the Keithley Direct I/O Library
Remove check from the 'Generate Component Wrapper' check box.
Push the Create Unit button.

Several files will be put into the imports folder (\Program Files\Borland\CBuilder5\Imports\). The important ones for this topic are the KPCIDIOLib_TLB.h and KPCIDIOLib_TLB.cpp files.

Now that we have wrapper files, we can begin coding.

The screen shots below show some specific code for a project. In this case the two wrapper files were relocated to the same folder as the project.

Notice a USEUNIT clause has been added to the application's cpp file:

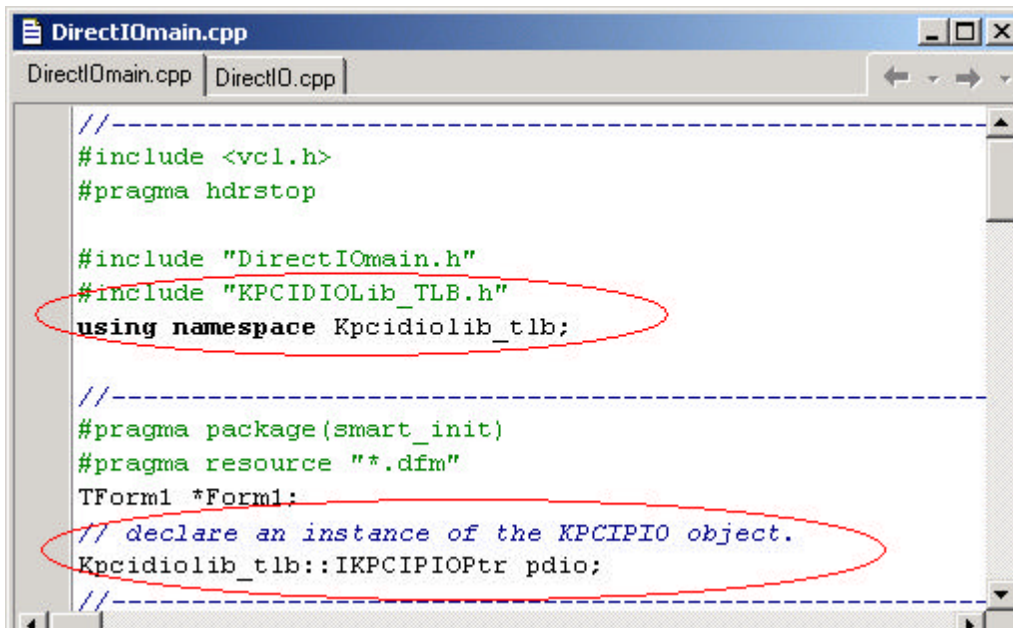


```
DirectIO.cpp
DirectIOmain.cpp DirectIO.cpp

//-----
#include <vc1.h>
#pragma hdrstop
USERES("DirectIO.res");
USEFORM("DirectIOmain.cpp", Form1);
USEUNIT("KPCIDIOLib_TLB.cpp");
//-----
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
    try
    {
        Application->Initialize();
        Application->CreateForm(__classid(TForm1), &Form1);
        Application->Run();
    }
    catch (Exception &exception)
    {
        Application->ShowException(&exception);
    }
    return 0;
}

IC++ Error DirectIOmain.cpp(31): E2379 Statement missing :
23: 16 Insert
```

Declarations in our source cpp file :



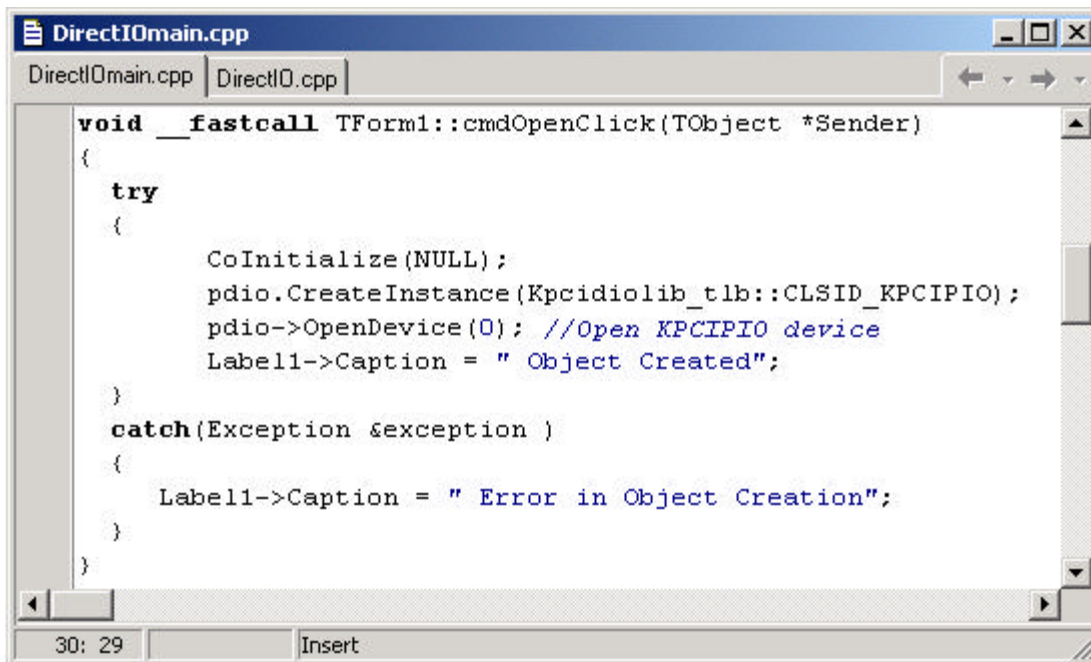
```
DirectIOmain.cpp
DirectIO.cpp

//-----
#include <vcl.h>
#pragma hdrstop

#include "DirectIOmain.h"
#include "KPCIDIOLib_TLB.h"
using namespace Kpcidiolib_tlb;

//-----
#pragma package (smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
// declare an instance of the KPCIPIO object.
Kpcidiolib_tlb::IKPCIPIOPtr pdio;
//-----
```

Instantiate the Object and Call OpenDevice method:



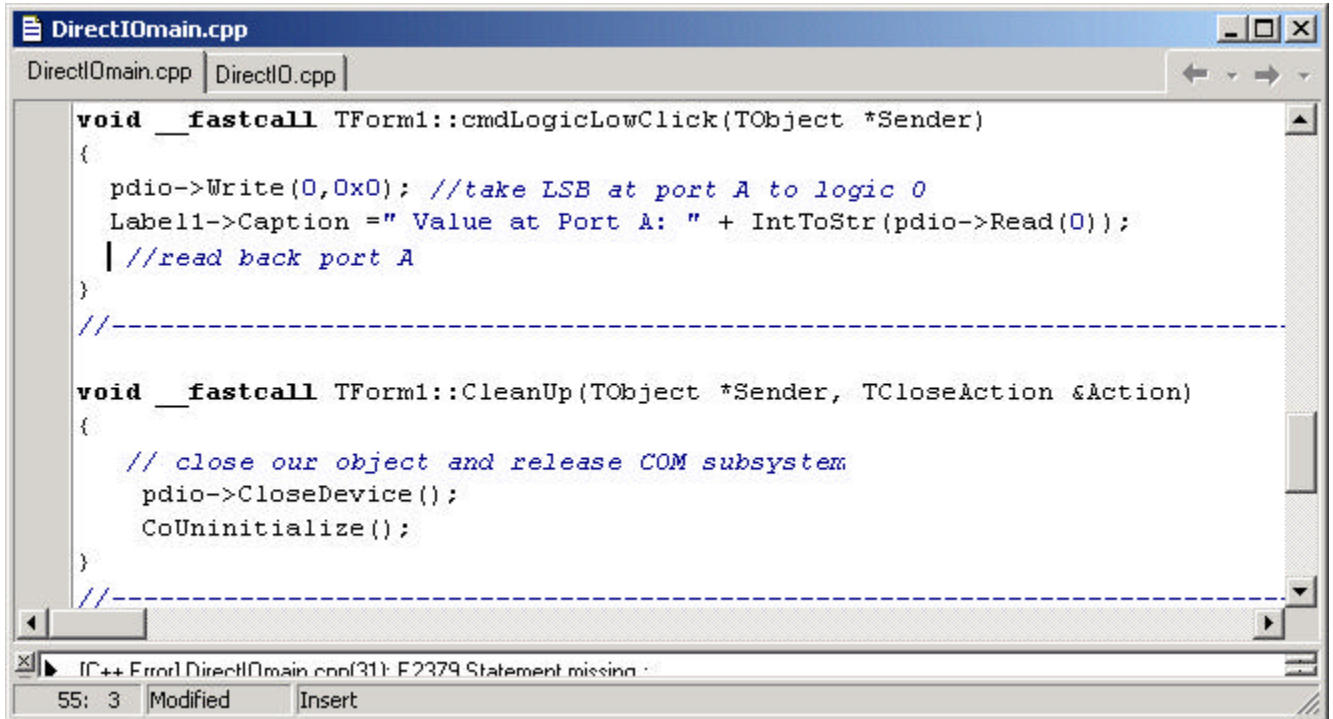
```
DirectIOmain.cpp
DirectIO.cpp

void __fastcall TForm1::cmdOpenClick(TObject *Sender)
{
    try
    {
        CoInitialize(NULL);
        pdio.CreateInstance(Kpcidiolib_tlb::CLSID_KPCIPIO);
        pdio->OpenDevice(0); //Open KPCIPIO device
        Label1->Caption = " Object Created";
    }
    catch(Exception &exception )
    {
        Label1->Caption = " Error in Object Creation";
    }
}

30: 29      Insert
```

Use the methods of the object and close on exit:

(Note: the code below does not show the port configuration step, e.g., `pdio->Write(3,0x80)`; to make all ports outputs prior to any attempt to write values to the ports)



```
DirectIOmain.cpp
DirectIO.cpp

void __fastcall TForm1::cmdLogicLowClick(TObject *Sender)
{
    pdio->Write(0,0x0); //take LSB at port A to logic 0
    Label1->Caption =" Value at Port A: " + IntToStr(pdio->Read(0));
    | //read back port A
}
//-----

void __fastcall TForm1::Cleanup(TObject *Sender, TCloseAction &Action)
{
    // close our object and release COM subsystem
    pdio->CloseDevice();
    CoUninitialize();
}
//-----

C++ Error DirectIOmain.cpp(31): E2379 Statement missing:
55: 3 Modified Insert
```

Conclusion: The Direct I/O COM API of DriverLINX provides a simple and fast interface to the hardware. With only a few steps, it can be used from the Borland environment.