SCSI TOOLBOX, LLC

Developer Toolbox API - Getting Started

Sales: 720.249.2641
General: 303.972.2072

Contents

Sales: 720.249.2641
General: 303.972.2072

Getting Started with DTB and Visual Studio in C++

In this document we will cover the basics of getting started with ScsiToolbox's Developer's Tool Box (DTB). The focus of the document will be to show you how to create an application using DTB – in particular how to create an application using VCPSSL.dll.

Checklist before starting:

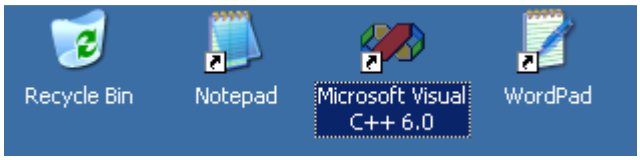
Prior to getting started, you will have to have done the following items:

- Installed SCSIToolbox
- Installed Visual Studio V++

OK, so let's get started. Our first project will be done with Visual Studio 6. After this, we will do the equivalent tasks in Visual Studio 2010 C++.

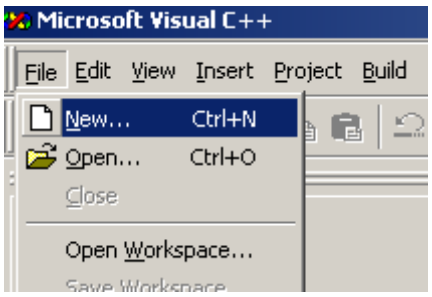
Create the Project:

Go ahead and launch Visual Studio 6 from your desktop as in the picture below:

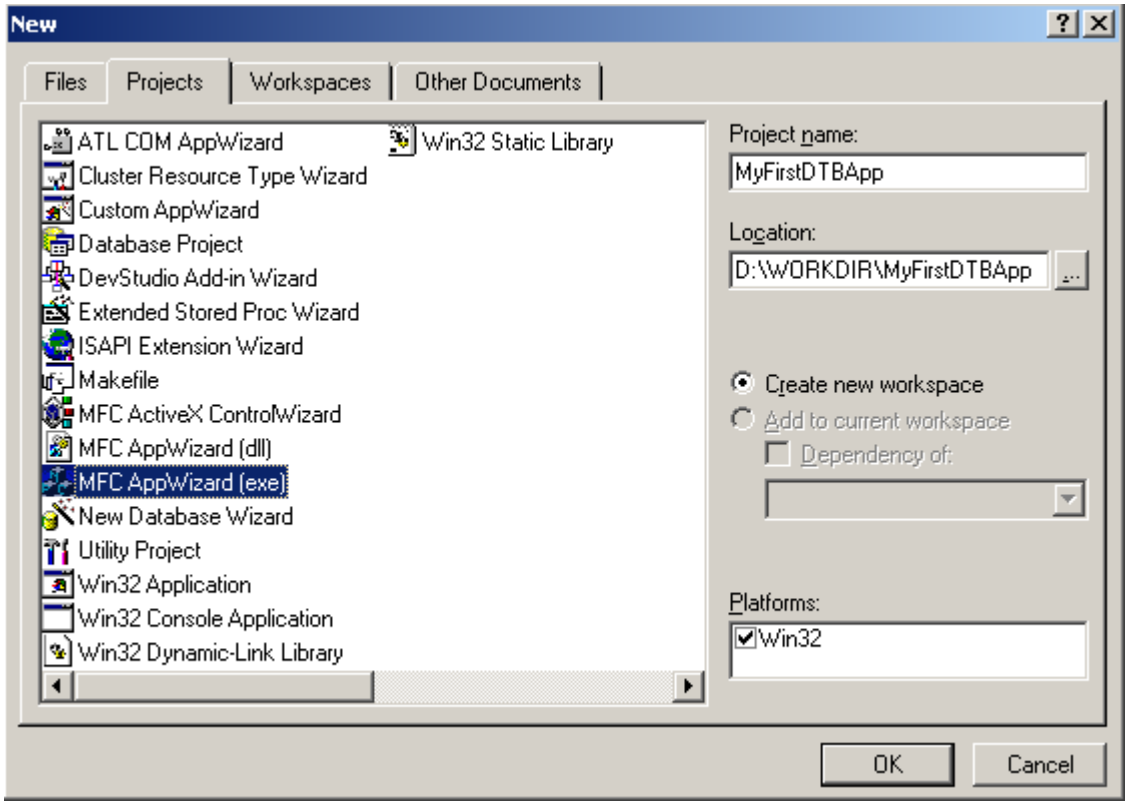


Now click on the menu item "File" => "New" (see the picture below):

Sales: 720.249.2641
General: 303.972.2072

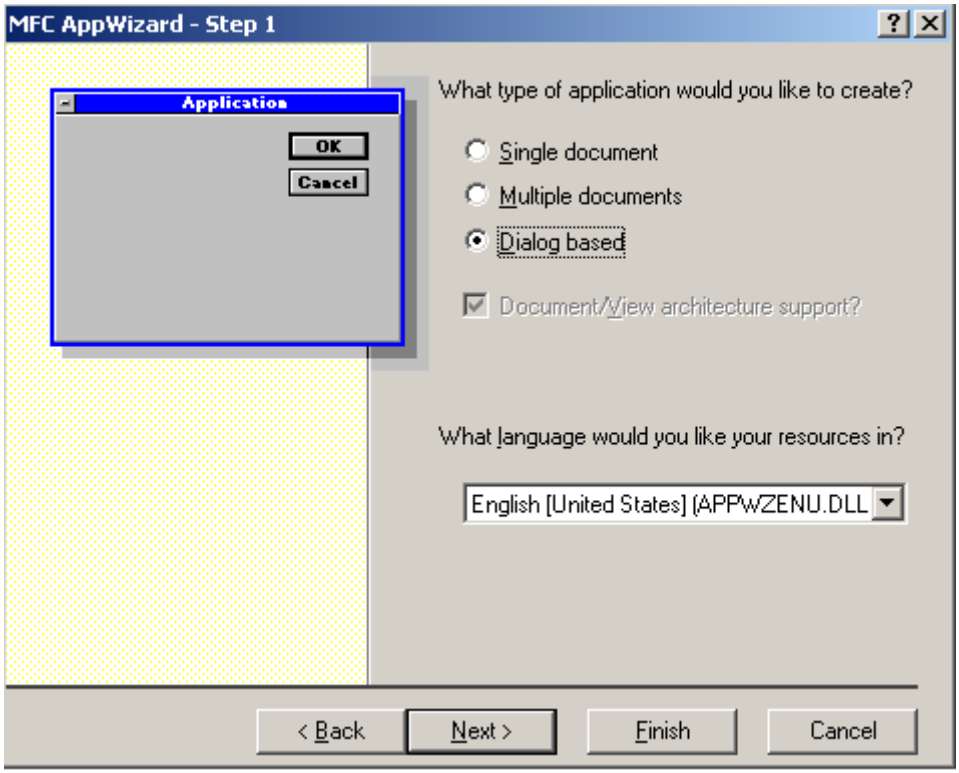


A tabbed dialog will appear (see the picture below). Select the “Projects” tab and then select the “MFC AppWizard (exe)” selection. On the right-hand side of the dialog, enter a name for the project (you can call it MyFirstDTBApp if you’d like) and then click “OK”.



Sales: 720.249.2641
General: 303.972.2072

After clicking the “OK” button, the following dialog will appear:

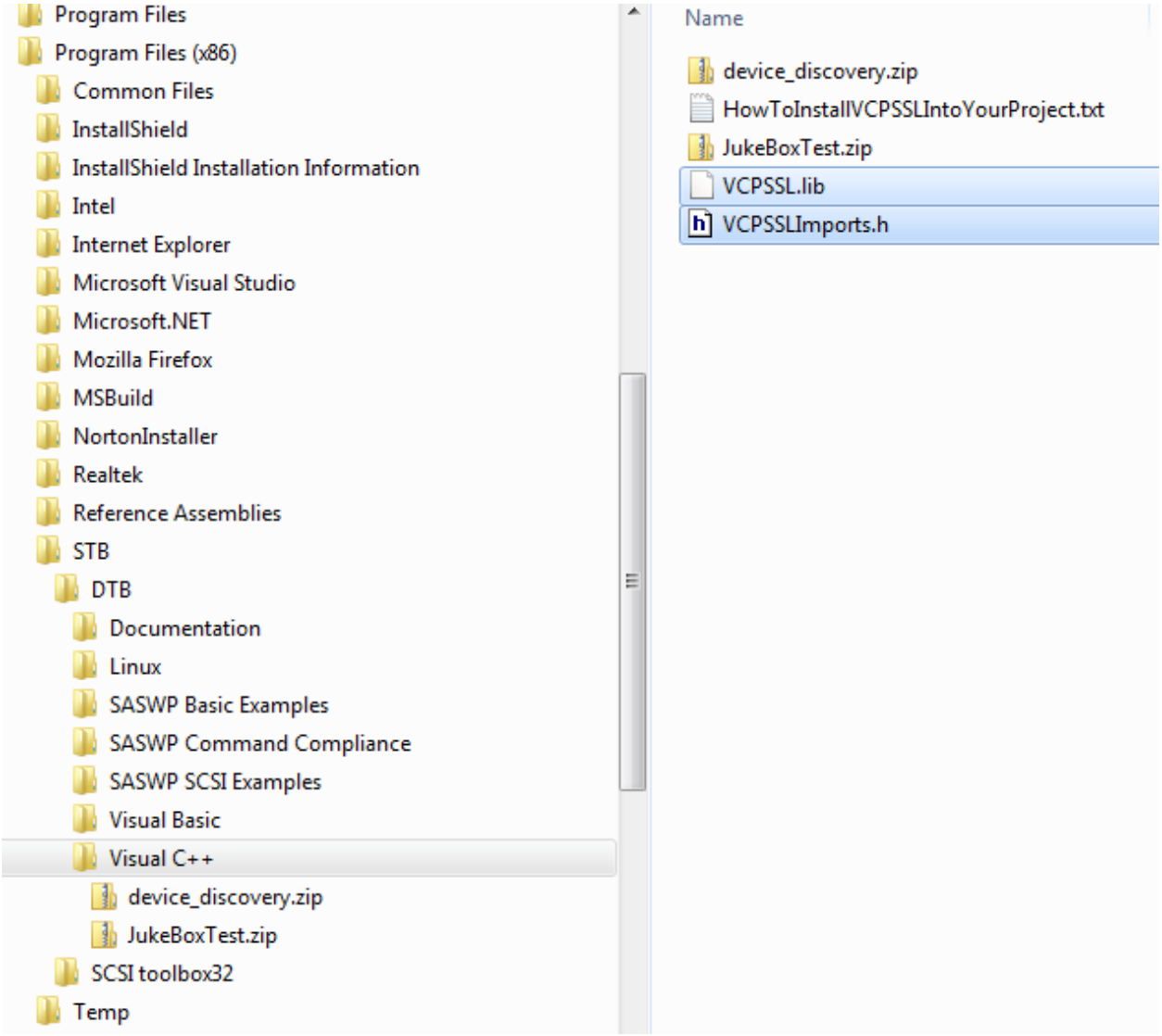


Select the “Dialog based” choice, and then click the “Finish” button.

Insert DTB files into your Project:

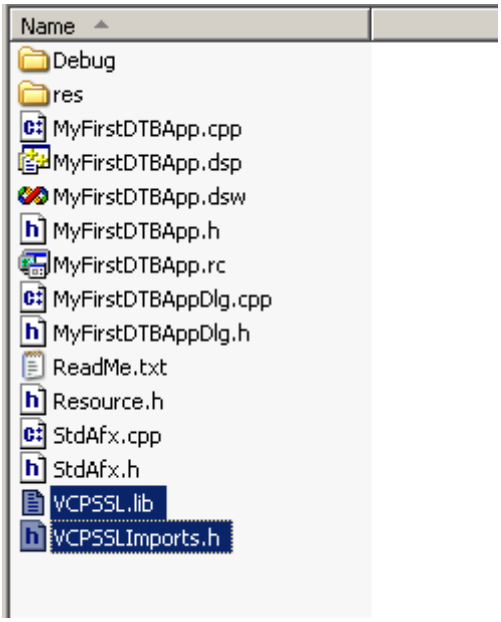
Certain files that SCSIToolbox supplies will need to be incorporated into your project. There are only 2 files: VCPSSLImports.h and VCPSSL.lib. These 2 files are located in the folder c:\Program Files\STB\DTB\VisualC++ (see picture below). On a 64-bit Operating System, the files will be in folder c:\Program Files (x86)\STB\DTB\Visual C++.

Sales: 720.249.2641
General: 303.972.2072

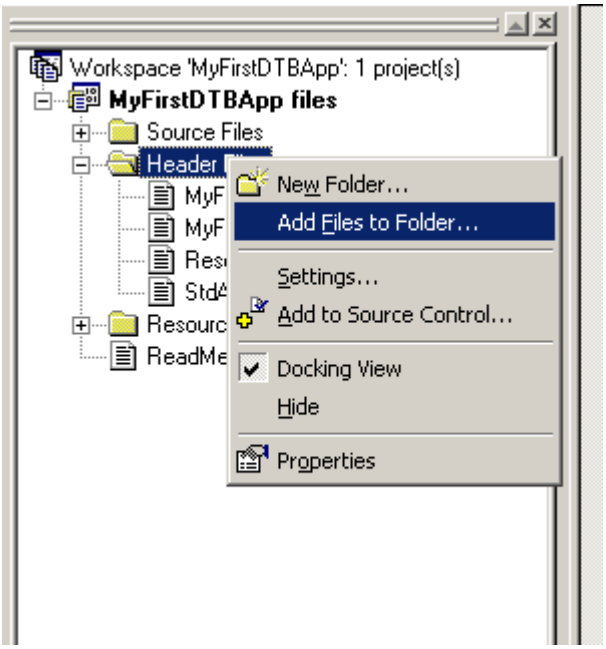


Copy these 2 files and store them into your project folder (see picture below).

Sales: 720.249.2641
General: 303.972.2072

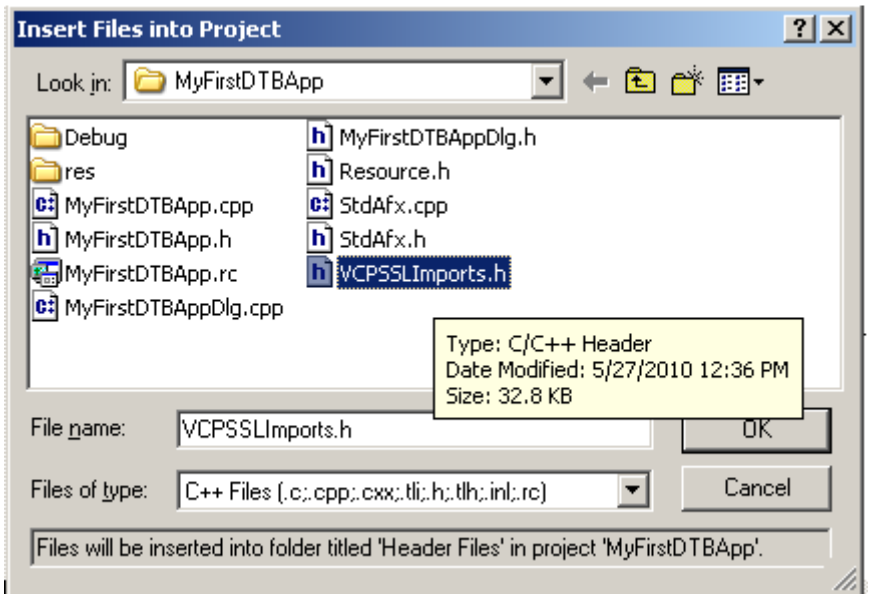


Back in Visual Studio, on the left-hand side you should see the “FileView” tab – click on it. Right-click on the “Header Files” and click “Add Files to Folder …” (see picture below).

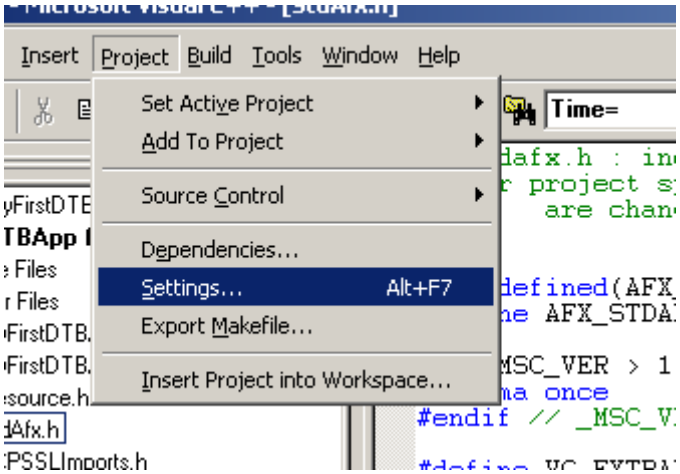


Sales: 720.249.2641
General: 303.972.2072

The “Insert Files into Project” dialog will appear. Insert the file “VCPSSLImports.h” file (see the picture below):

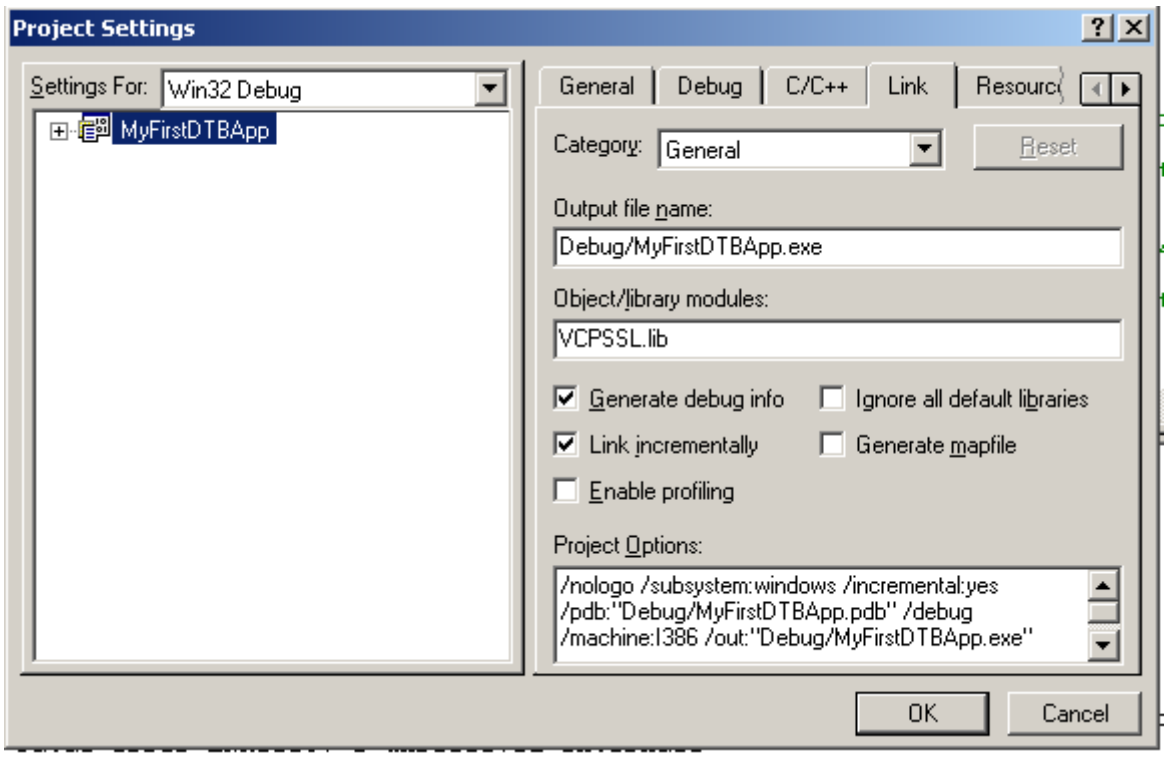


Now go to menu Project => Settings



and open up the “Link” tab and type in “VCPSSL.lib” in the “Imports” box.

Sales: 720.249.2641
General: 303.972.2072



Advanced Point: What this does is makes all the exported functions in the library VCPSSL.dll “visible” to your application.

Now open up the file StdAfx.h and type in the line

#include “VCPSSLImports.h”

Sales: 720.249.2641
General: 303.972.2072

```
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define VC_EXTRALEAN    // Exclude rarely-used stuff from Windows heac

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions
#include <afxdisp.h>        // MFC Automation classes
#include <afxdtctl.h>   // MFC support for Internet Explorer 4 Common
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>     // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT


//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediatel

#include "VCPSSLImports.h"

#endif // !defined(AFX_STDAFX_H__340509EA_6B4D_490E_96C6_0E3647A98553_
```

“Including” VCPSSLImports.h” into your project makes all the APIs in the VCPSSL.dll available to your project.

We are now done with adding all the necessary files into our project.

Build Your Project:

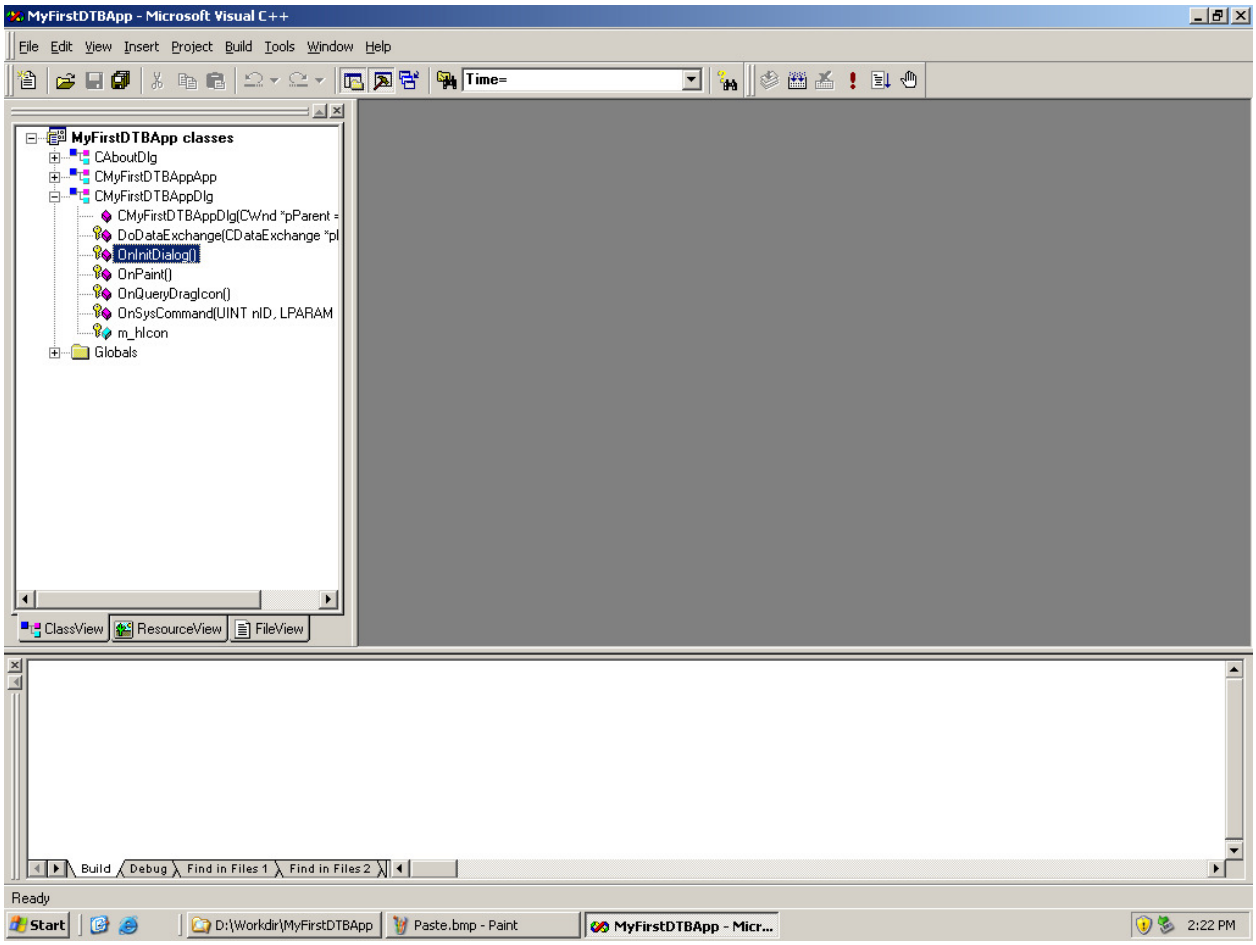
Just to make sure you’ve done everything correct so far, go ahead and build the project by clicking the “Build” button (in the picture below, the “build” button is the second one on the left):

Did the build work?

Use your first API from VCPSSL.dll:

Sales: 720.249.2641
General: 303.972.2072

We're now ready to call our first API from the VCPSSL library. We will call the API VCSCSIGetDLLVersion API to get the version of the VCPSSL library. Locate the function OnInitDialog:



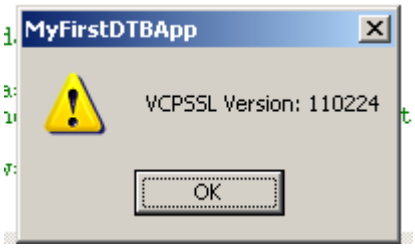
and at the very end of the function add the lines

```
int nVersion = VCSCSIGetDLLVersion( );

CString strInfo;

strInfo.Format("VCPSSL Version: %d",nVersion);

AfxMessageBox(strInfo);
```

```
    // Set the icon for this dialog.  The framework does this automatically
    //  when the application's main window is not a dialog
    SetIcon(m_hIcon, TRUE);     // Set big icon
    SetIcon(m_hIcon, FALSE);    // Set small icon

    // TODO: Add extra initialization here
    int nVersion = VCSCSIGetDLLVersion();
    CString strInfo;
    strInfo.Format("VCPSSL Version: %d",nVersion);
    AfxMessageBox(strInfo);

    return TRUE;  // return TRUE  unless you set the focus to a control
}

void CMyFirstDTBAppDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
    if ((nID & 0xFFF0) == IDM_ABOUTBOX)
    {
```

Now re-build your project and launch it by clicking the button “Launch” (in the picture below the “Launch” button is the fourth one on the left and is the “Exclamation Point”).

Do you get a message box like the one below?



Hopefully you’ve been able to follow along so far. At this point we will add code to our project to display all the devices on your system.

Sales: 720.249.2641
General: 303.972.2072

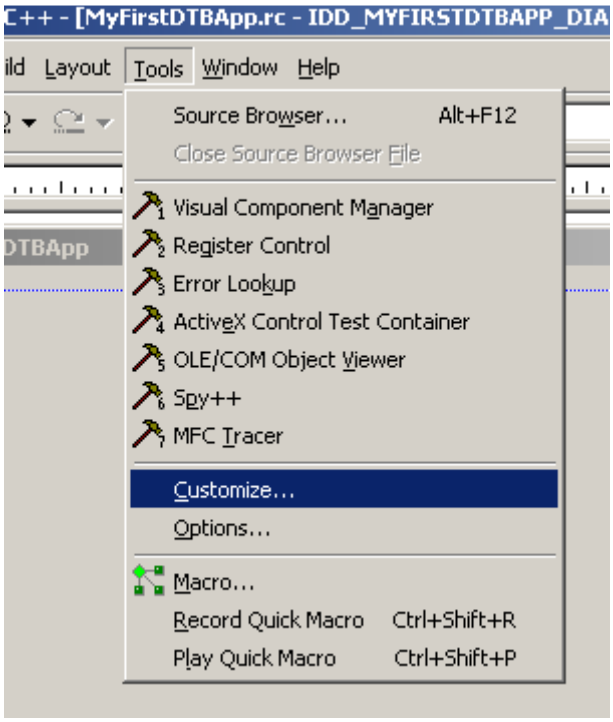
Display ALL Devices on your System:

We will continue with the MyFirstDTBApp project and extend it so that it displays all devices on your system. We will have to add a listbox to our display, and then add some API calls to retrieve the devices on the system. Let's get started by adding a listbox to our display.

Bring up the "Controls Toolbar":

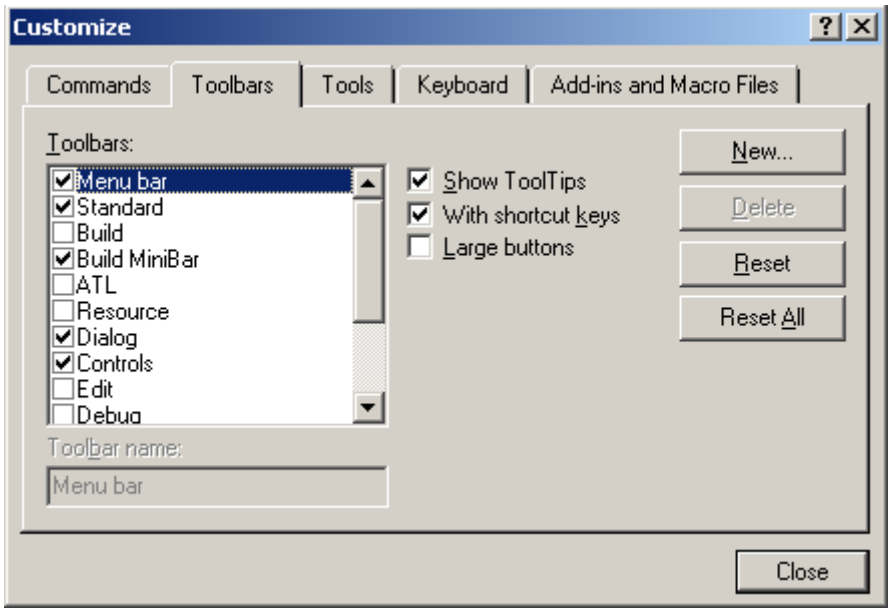
If you don't have the "Controls Toolbar" (this is a toolbar which has display buttons with all the controls available to your project), then follow these instructions:

Go to menu "Tools" => "Customize" (see the picture below)



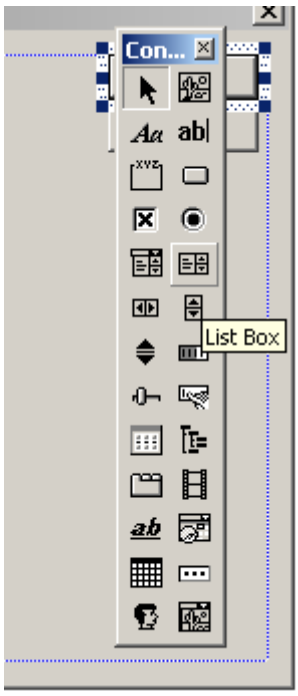
Sales: 720.249.2641
General: 303.972.2072

And then click on “Toolbars” tab, and then select “Controls” (see the picture below):



At this point the “Controls Toolbar” will be visible (your “Controls Toolbar” should look very similar to the picture below).

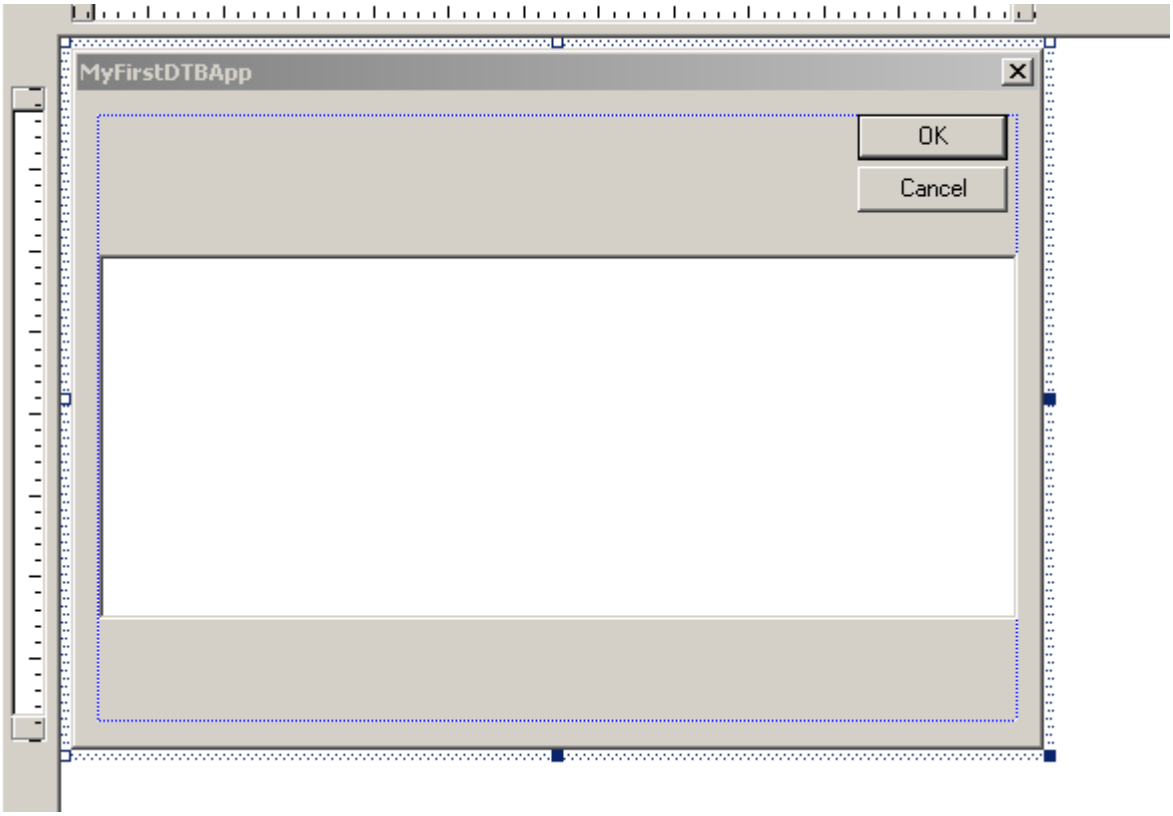
Sales: 720.249.2641
General: 303.972.2072



Insert a "Listbox" to your display:

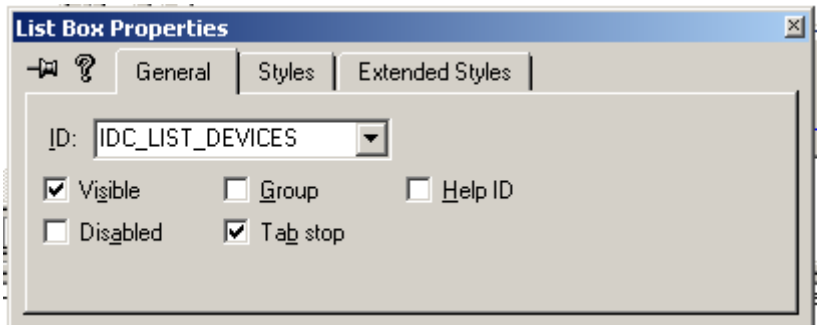
From the "Controls Toolbar" click on the "List Box" icon and then "drag" it to your dialog. Your dialog should now look as in the picture below:

Sales: 720.249.2641
General: 303.972.2072



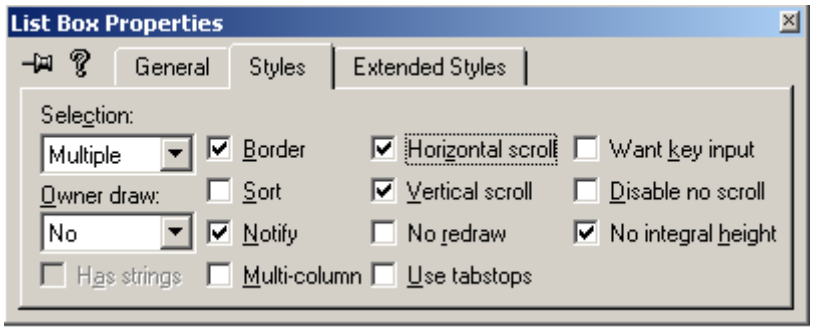
The listbox that we just added is the “middle white area” in the picture above.

Each control on your display has a Control ID – we are going to change this ID. To do this right-click on the listbox that is on your display and click “Properties”. For the “ID:” field, change it from IDC_LIST1 to IDC_LIST_DEVICES as in the picture below:

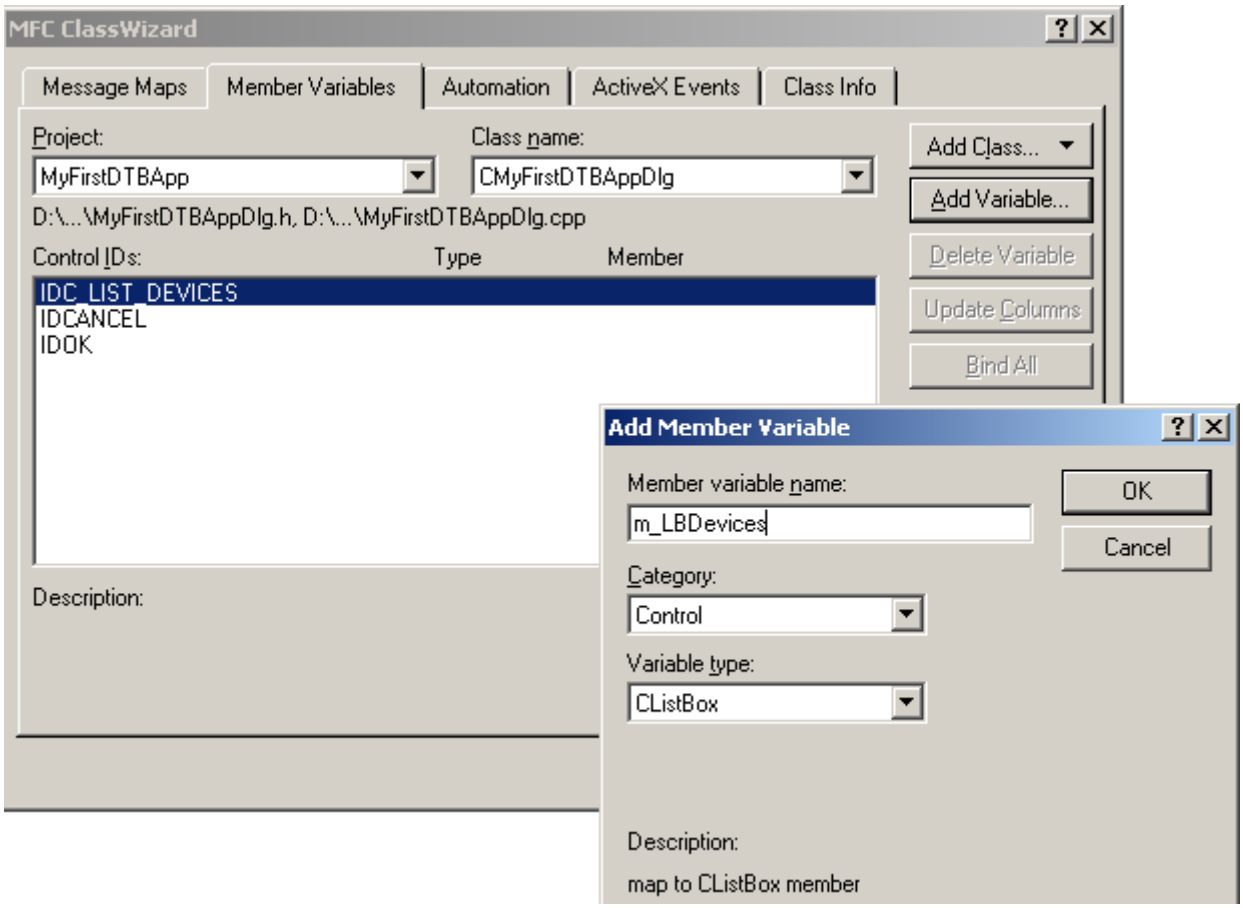


Sales: 720.249.2641
General: 303.972.2072

Now click on the “Styles” tab and make sure you uncheck “Sort”. Also make sure everything looks like the picture below:



Now we will add a variable that represents the listbox. To do this, back in the main window for your project, use menu item “View” => “ClassWizard”. Click on the “Member Variables” tab, and select the “IDC_LIST_DEVICES” control ID, and then click the “Add Variable…” button. You should see a dialog like the one in the picture below:



Go ahead and fill in the "Add Member Variable" as you see in the picture above – that is make the variable name "m_LBDevices", and select "Category" to be "Control". Click on the "OK" button.

The above steps added a listbox control to our display. That may have seemed like a lot of work, but after you do it several times you should be able to accomplish these steps in only a few minutes.

Add Code to Display Devices on your System:

OK, now we're ready to add the necessary code to display all the devices on your system. The basic outline of this code is that first we will call an API to retrieve the number of Host Bus Adapters (HBA) on your system, and then for each HBA

we call an API to tell us the number of maximum possible targets on this HBA. Then for luns 0 thru 7, we call an API to tell us the device type of the HBA:Target:Lun address. If there is a device at this address, we then retrieve the Vendor, the Product, and the firmware Version of the device.

Bring up the code in OnInitDialog where we called API VCSCSIGetDLLVersion. After that code that you added earlier, append the following code (as in the picture below):

```
int nNumTargetsOnHBA;
int nDeviceType;
char cVendor[128] = {0};
char cProduct[128] = {0};
char cVersion[128] = {0};

int nNumHBA = VCSCSIHostAdapterCount();
for (int nHA = 0;nHA < nNumHBA;nHA++)
{
  nNumTargetsOnHBA = VCSCSITargetCount(nHA);
  for (int nTid = 0;nTid < nNumTargetsOnHBA;nTid++)
  {
    for (int nLun = 0;nLun < 8;nLun++)
    {
      nDeviceType = VCSCSIGetDeviceType(nHA,nTid,nLun);
      if (nDeviceType >= 0 && nDeviceType <= 31)
      {
        VCSCSIGetVendor(nHA,nTid,nLun,&cVendor[0]);
        VCSCSIGetProduct(nHA,nTid,nLun,&cProduct[0]);
        VCSCSIGetVersion(nHA,nTid,nLun,&cVersion[0]);
        strInfo.Format("At address %d:%d:%d is device type %d, Vendor=%s, Product=%s, Version=%s",
                       nHA,nTid,nLun,nDeviceType,cVendor,cProduct,cVersion);
        m_LBDevices.AddString(strInfo);
      }
    }
  }
}
```

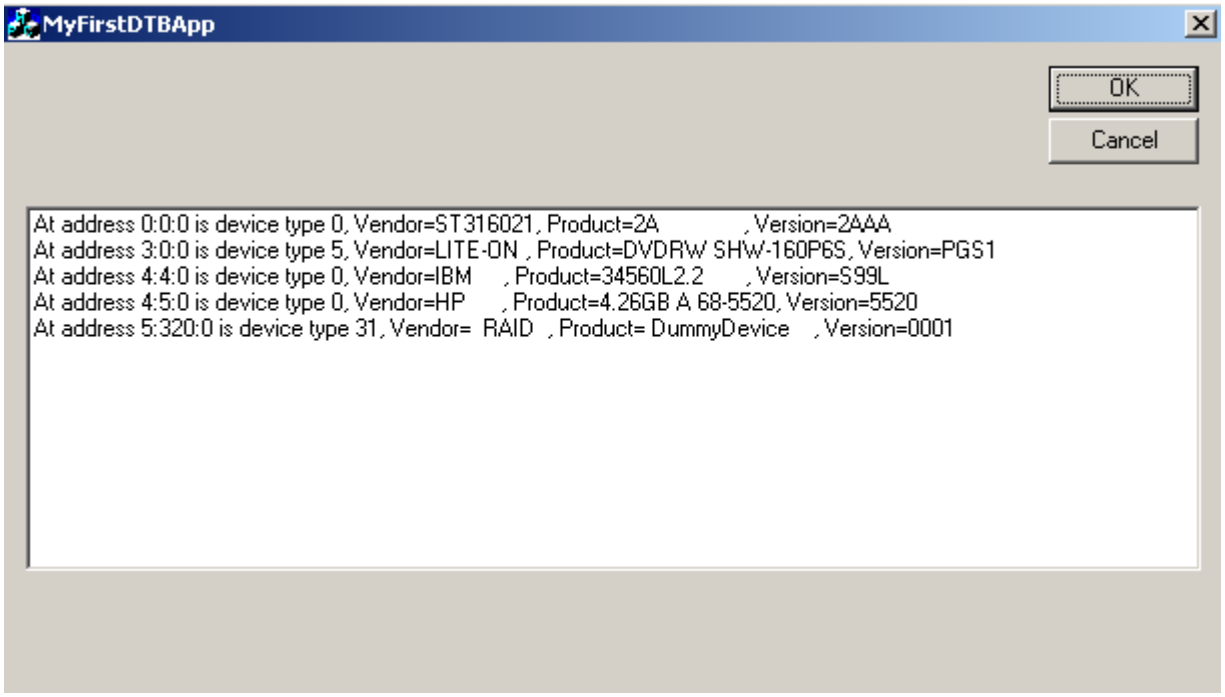
We also include the code below so that you can “copy-and-paste” it into your project:

```
int nNumTargetsOnHBA;
int nDeviceType;
char cVendor[128] = {0};
char cProduct[128] = {0};
char cVersion[128] = {0};

int nNumHBA = VCSCSIHostAdapterCount();
for (int nHA = 0;nHA < nNumHBA;nHA++)
{
  nNumTargetsOnHBA = VCSCSITargetCount(nHA);
  for (int nTid = 0;nTid < nNumTargetsOnHBA;nTid++)
```


Sales: 720.249.2641
General: 303.972.2072

```
    {
      for (int nLun = 0;nLun < 8;nLun++)
      {
        nDeviceType = VCSCSIGetDeviceType(nHA,nTid,nLun);
        if (nDeviceType >= 0 && nDeviceType <= 31)
        {
          VCSCSIGetVendor(nHA,nTid,nLun,&cVendor[0]);
          VCSCSIGetProduct(nHA,nTid,nLun,&cProduct[0]);
          VCSCSIGetVersion(nHA,nTid,nLun,&cVersion[0]);
          strInfo.Format("At address %d:%d:%d is device type %d,
Vendor=%s, Product=%s, Version=%s",

nHA,nTid,nLun,nDeviceType,cVendor,cProduct,cVersion);
          m_LBDevices.AddString(strInfo);
        }
      }
    }
  }
```

Finally, build your application and launch it – you should have a display similar to the picture below:



Sales: 720.249.2641
General: 303.972.2072

In this document we utilized only 7 APIs in VCPSSL.dll; currently there are over 300 APIs available!!! You can accomplish almost anything you want with the full range of the APIs!

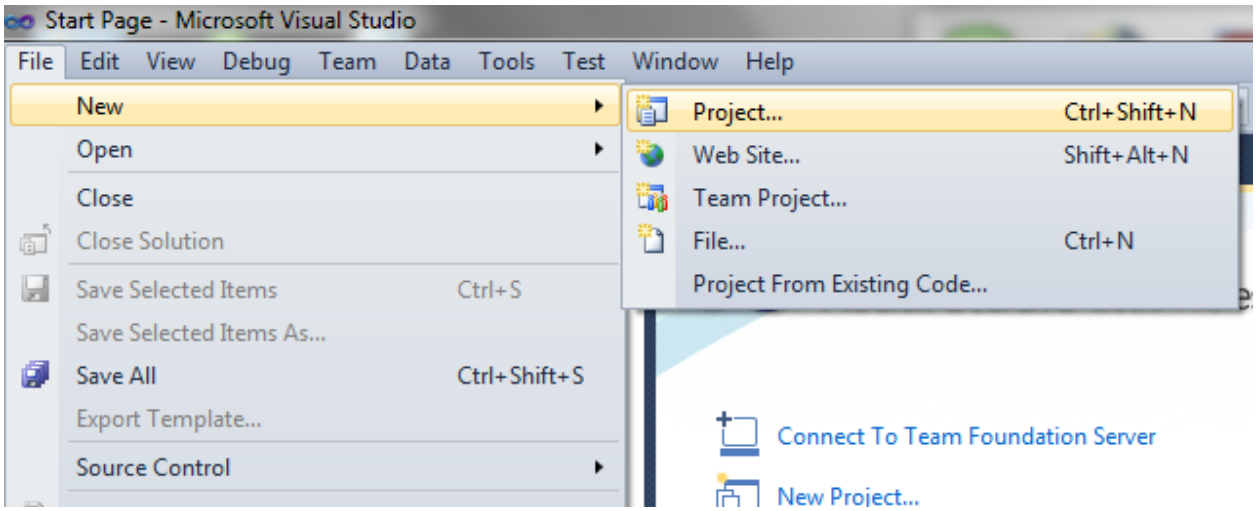
Getting Started in Visual Studio 2010:

A lot of the steps are similar in Visual Studio 2010 as they are in Visual Studio 6. But there are differences, so we're going to cover mainly the areas that are different. The steps are basically the same, but how you accomplish each step may be different.

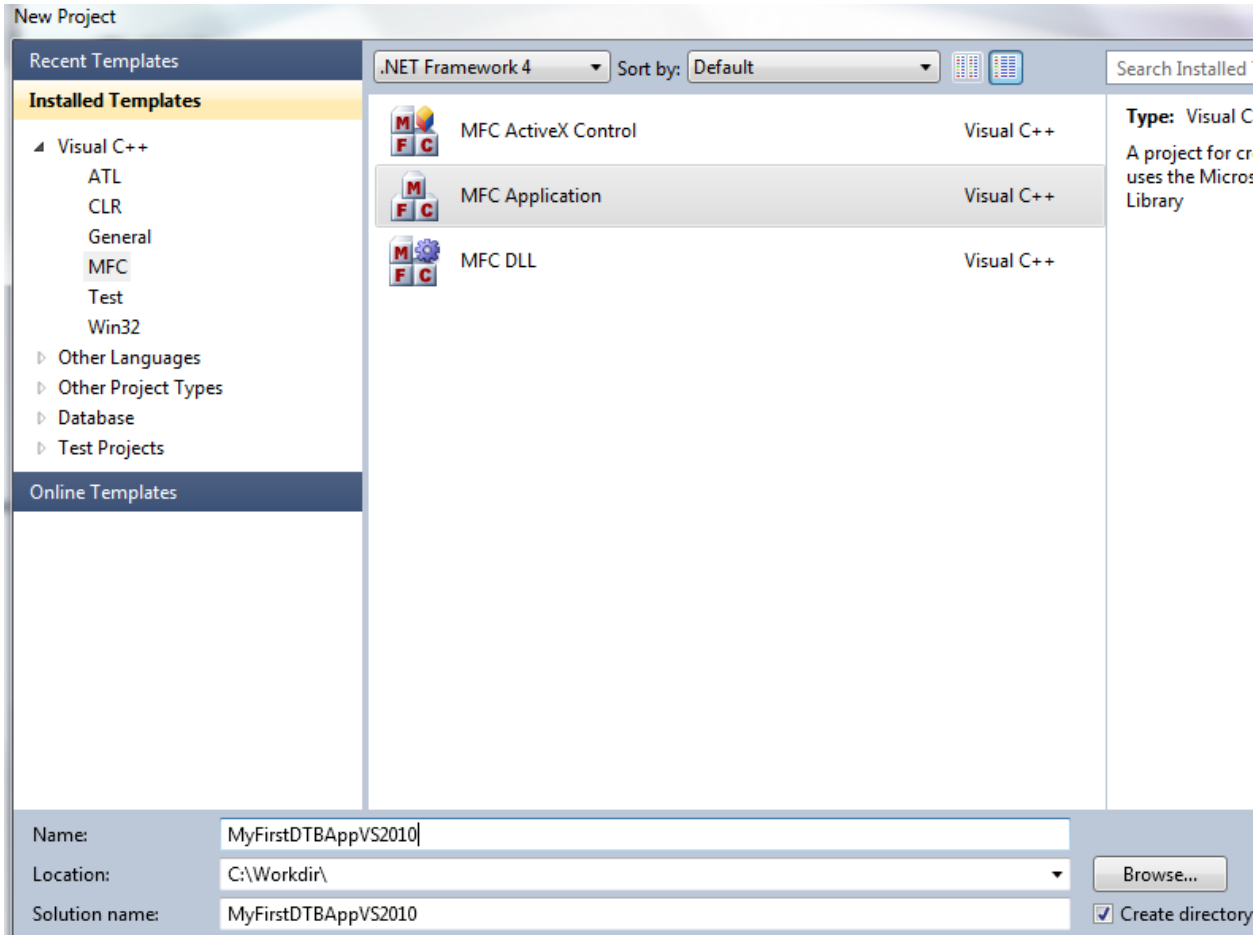
As in VS6, we start by launching Visual Studio 2010 from your desktop:

And from the menus at the top, click on "File" => "New" => "Project …"

Sales: 720.249.2641
General: 303.972.2072

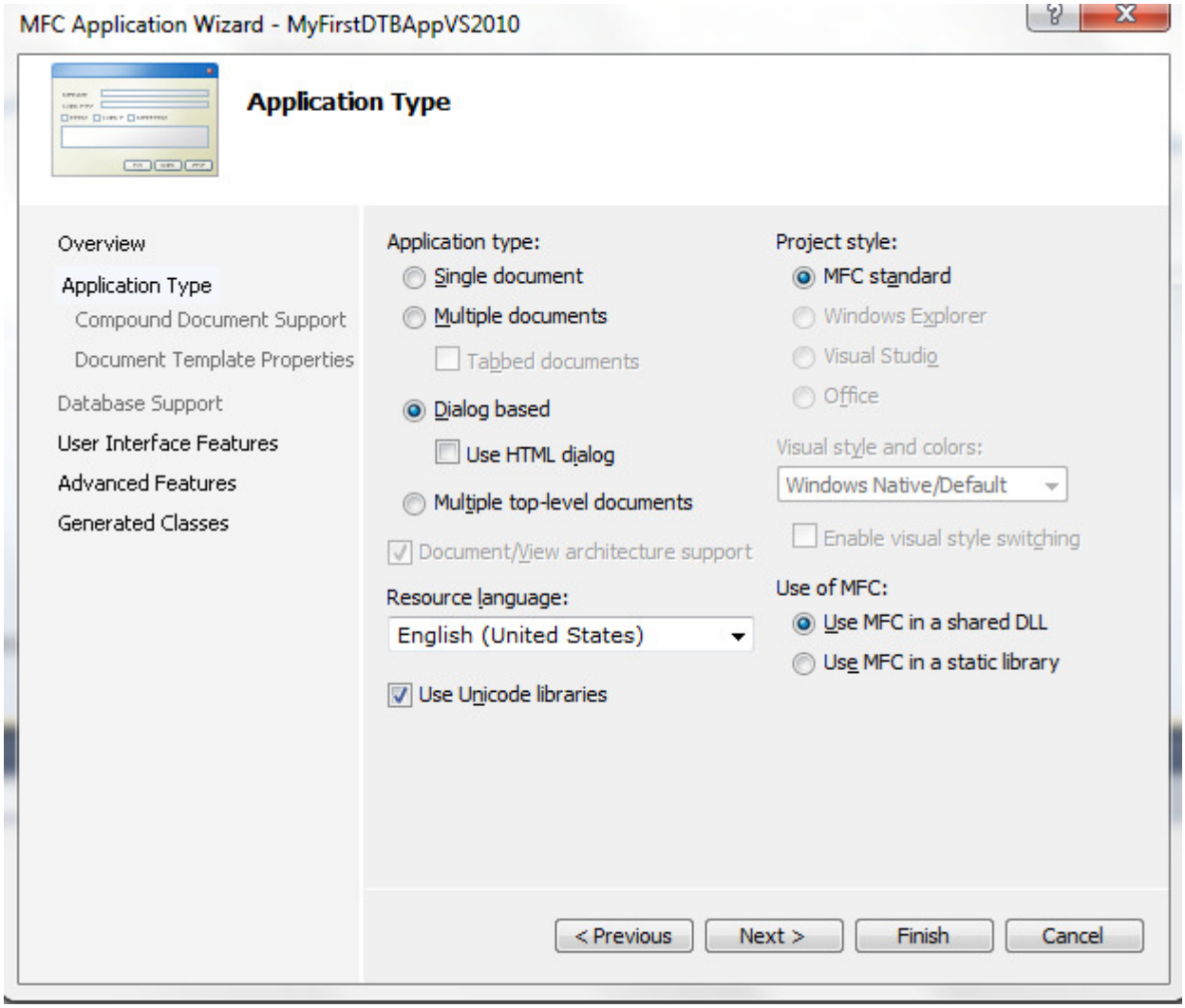


Open up the Visual C++ folder, and then select “MFC Application” (see picture below). Go ahead and call the project “MyFirstDTBAppVS2010” as you see in the picture below:



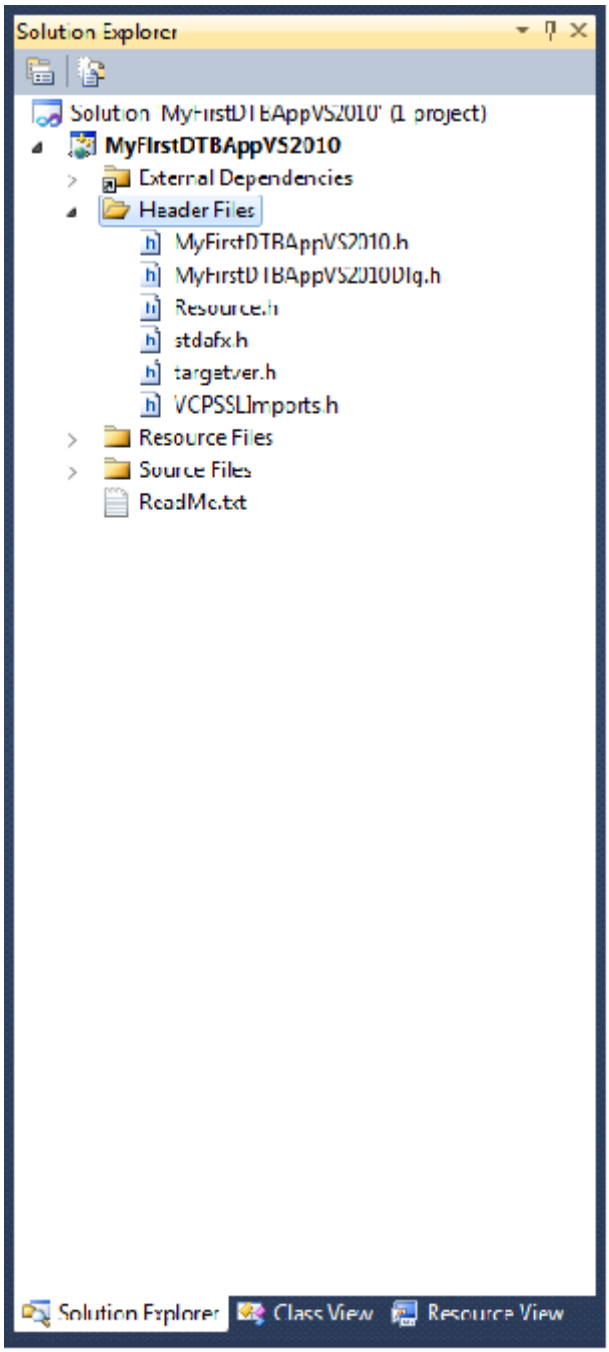
The “Application Type” dialog will appear (see picture below). Select “Dialog based” and then click the “Finish” button.

Sales: 720.249.2641
General: 303.972.2072



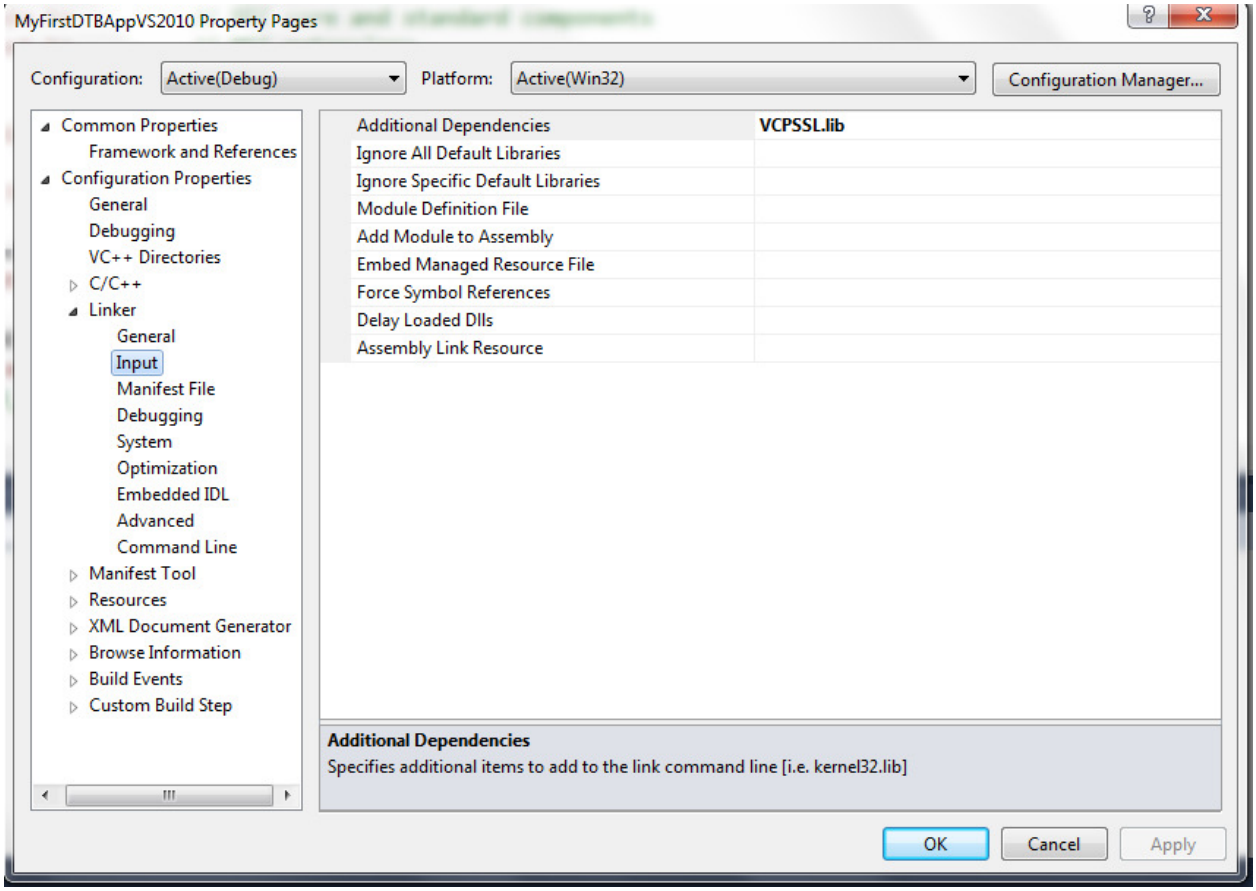
As in VS6, copy the 2 files VCPSSLImports.h and VCPSSL.lib from your c:\Program Files\STB\DTB\Visual C++ folder, and copy them into your project folder.

Now insert VCPSSLImports.h into your “Header Files” project folder (in “Solution Explorer” on the left hand side, right-click on the “Header Files” folder, and click “Add” => “Existing Item” and then browse to where “VCPSSLImports.h” is located.



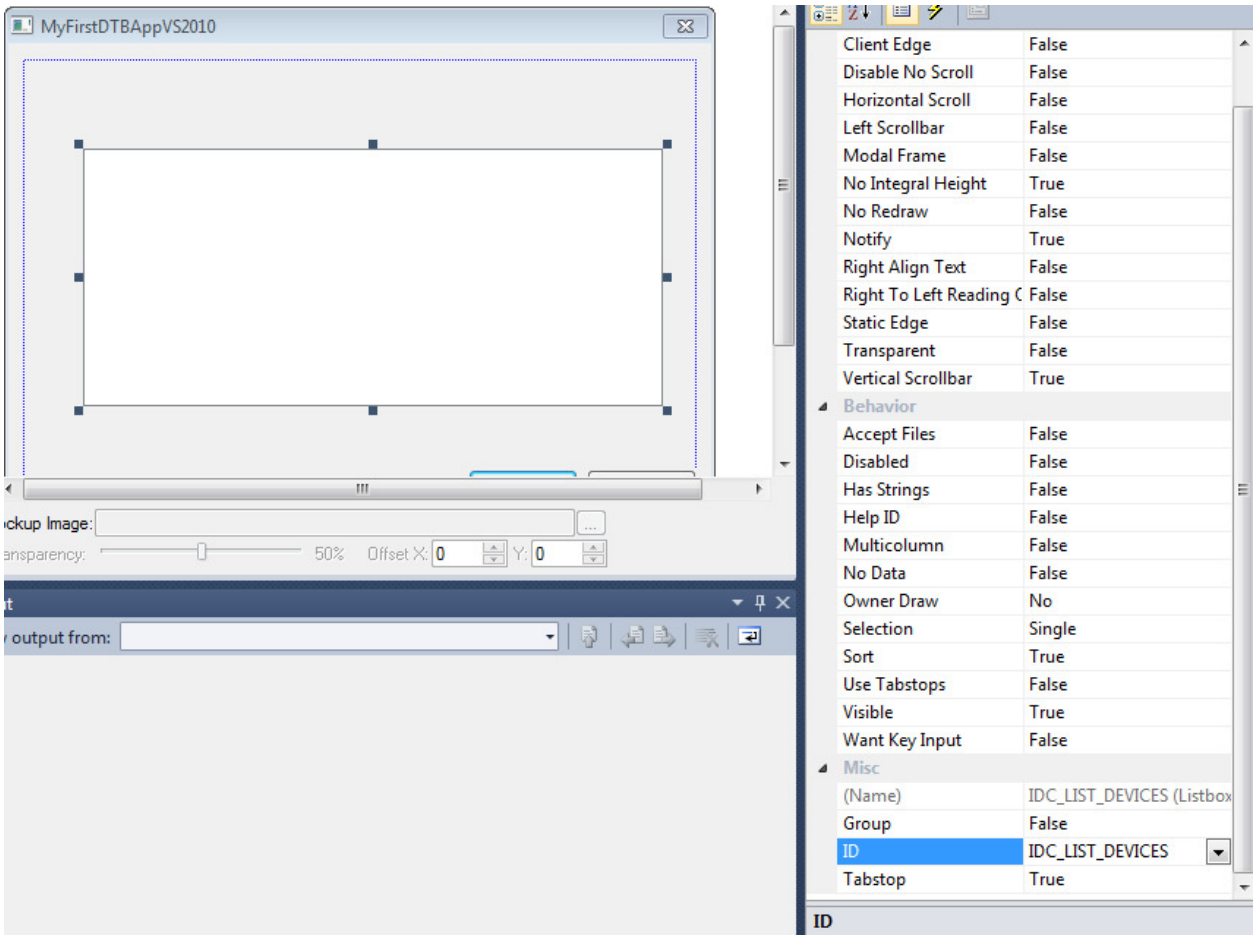
Sales: 720.249.2641
General: 303.972.2072

Now we insert VCPSSL.lib into our project. At the main menu items, click on "Project" and then "MyFirstDTBAppVS2010 Properties …". The following dialog will appear:

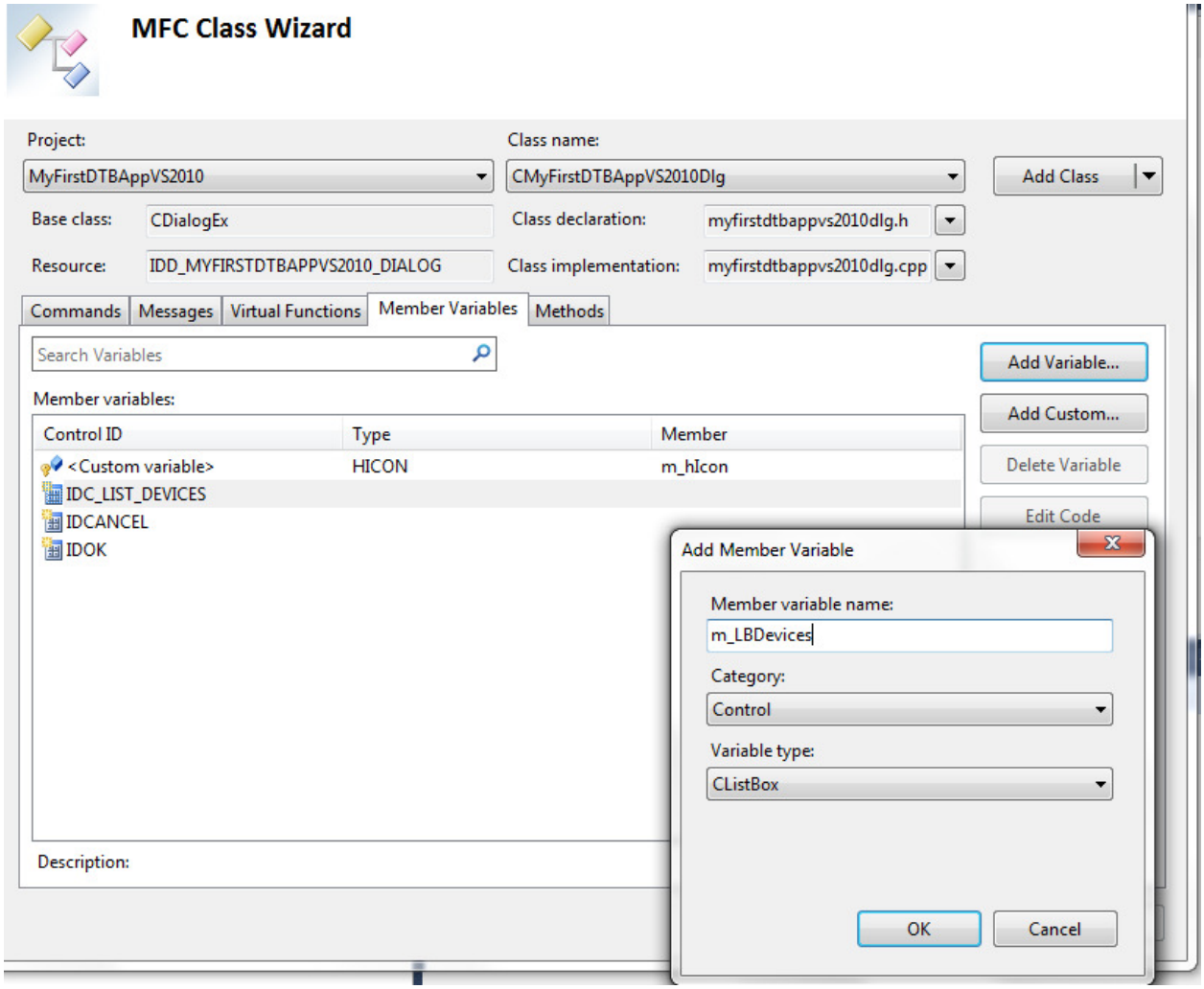


As in the picture above, open up the "Configuration Properties" folder, then the "Linker" folder, and then click on the "Input" folder. On the right-hand side of the dialog, where the "Additional Dependencies" is located, input "VCPSSL.lib" as you see in the above picture.

Now we add the listbox to our dialog. Just as in our VS6 project, add a listbox to your dialog and change the ID of the listbox to IDC_LIST_DEVICES (see the picture below):



And now add a member variable for this listbox: Do this by going to menu "Project" => "ClassView" (see the picture below):



In the above picture, you see the “MFC ClassWizard” dialog has been displayed. Click on the “Member Variables” tab, select your ID “IDC_LIST_DEVICES”, and then click the “Add Variable …” button. In the “Add Member Variable” dialog, enter the information you see in the picture above.

OK, so we’re now ready to add the code. Just as we did in our VS6 project, bring up the code for OnInitDialog and add the following lines of code (this code is identical to that in our VS6 project so you can copy-n-paste it):

int nVersion = VCSCSIGetDLLVersion( );

CString strInfo;

strInfo.Format(“VCPSSL Version: %d”,nVersion);

AfxMessageBox(strInfo);

```
int nNumTargetsOnHBA;
int nDeviceType;
char cVendor[128] = {0};
char cProduct[128] = {0};
char cVersion[128] = {0};

int nNumHBA = VCSCSIHostAdapterCount();
for (int nHA = 0;nHA < nNumHBA;nHA++)
{
  nNumTargetsOnHBA = VCSCSITargetCount(nHA);
  for (int nTid = 0;nTid < nNumTargetsOnHBA;nTid++)
  {
    for (int nLun = 0;nLun < 8;nLun++)
    {
      nDeviceType = VCSCSIGetDeviceType(nHA,nTid,nLun);
      if (nDeviceType >= 0 && nDeviceType <= 31)
      {
        VCSCSIGetVendor(nHA,nTid,nLun,&cVendor[0]);
        VCSCSIGetProduct(nHA,nTid,nLun,&cProduct[0]);
        VCSCSIGetVersion(nHA,nTid,nLun,&cVersion[0]);
        strInfo.Format("At address %d:%d:%d is device type %d,
Vendor=%s, Product=%s, Version=%s",

nHA,nTid,nLun,nDeviceType,cVendor,cProduct,cVersion);
          m_LBDevices.AddString(strInfo);
      }
    }
  }
}
```

Now build your project by clicking menu “Build” => “Build Solution”.

Sales: 720.249.2641
General: 303.972.2072

Hopefully everything builds correctly! Congratulations – you've created your first DTB application in Visual Studio 2010.

Sales: 720.249.2641
General: 303.972.2072