Creating and Running 32-bit Windows Applications on 64-bit Windows
8 February 2013

Alta API Version 2.6.0.0 introduces the ability for developers to successfully create 32-bit Windows (Win32) applications on 64-bit Windows (x64) operating systems. Additionally, 32-bit Alta applications can now be successfully ported to the 64-bit OS by using the proper DLLs as described here.

This application note provides an example of how to create a 32-bit application for a 64-bit OS using the Alta API and an Alta-supplied example program. The example source used is **ADT_L1_1553_ex_bc2int.c**, found in the Alta distribution at
C:\Program Files\Alta Data Technologies\Alta Software\ADT_L1_API\examples\M1553 Examples.
This example sets up a simple BC (Bus Controller) on an Alta 1553 device that runs until stopped by the user and generates interrupt events to thoroughly exercise the API, device driver, and kernel plug-in.

Note: For customers using Alta ARINC products, simply substitute the ARINC example ADT_L1_A429_ex_rxtx1_int.c in place of the 1553 example mentioned throughout this app note. This ARINC example source is located at C:\Program Files\Alta Data Technologies\Alta Software\ADT_L1_API\examples\A429 Examples\.
This ARINC example also generates interrupt events to thoroughly exercise the API, device driver, and kernel plug-in, using an Alta ARINC product.

This example uses Microsoft Developer Studio 2008 but the usage of lib and DLL files can be applied to other development environments.

SKIP THE DETAILS

For users who want to skip over the details of creating a Win32 project on X64, here's what you need to know:

The Alta 64-bit API version 2.6.0.0 contains the standard distributed 64-bit Layer 0 and Layer 1 DLLs for creating 64-bit applications. Additionally, this API contains the 32-bit Layer 0 and Layer 1 DLLs for creating 32-bit applications *on 64-bit systems*. **NOTE: The 32-bit Layer 0 DLL included in the 64-bit Alta distribution was built on a 64-bit machine and is NOT the same as the 32-bit Layer 0 DLL built on a 32-bit machine.**

Create a 64-bit application on 64-bit OS:
Use these libraries and DLLs:
C:\Program Files\Alta Data Technologies\Alta Software\ADT_L0_API**Win64**\bin\ADT_L0.dll
C:\Program Files\Alta Data Technologies\Alta Software\ADT_L0_API**Win64**\bin\ADT_L0.lib
C:\Program Files\Alta Data Technologies\Alta Software\ADT_L1_API**Win64**\bin\ADT_L1.dll
C:\Program Files\Alta Data Technologies\Alta Software\ADT_L1_API**Win64**\bin\ADT_L1.lib

Create a 32-bit application on 64-bit OS:
Use these libraries and DLLs:
C:\Program Files\Alta Data Technologies\Alta Software\ADT_L0_API**x86**\bin\ADT_L0.dll
C:\Program Files\Alta Data Technologies\Alta Software\ADT_L0_API**x86**\bin\ADT_L0.lib
C:\Program Files\Alta Data Technologies\Alta Software\ADT_L1_API**x86**\bin\ADT_L1.dll
C:\Program Files\Alta Data Technologies\Alta Software\ADT_L1_API**x86**\bin\ADT_L1.lib

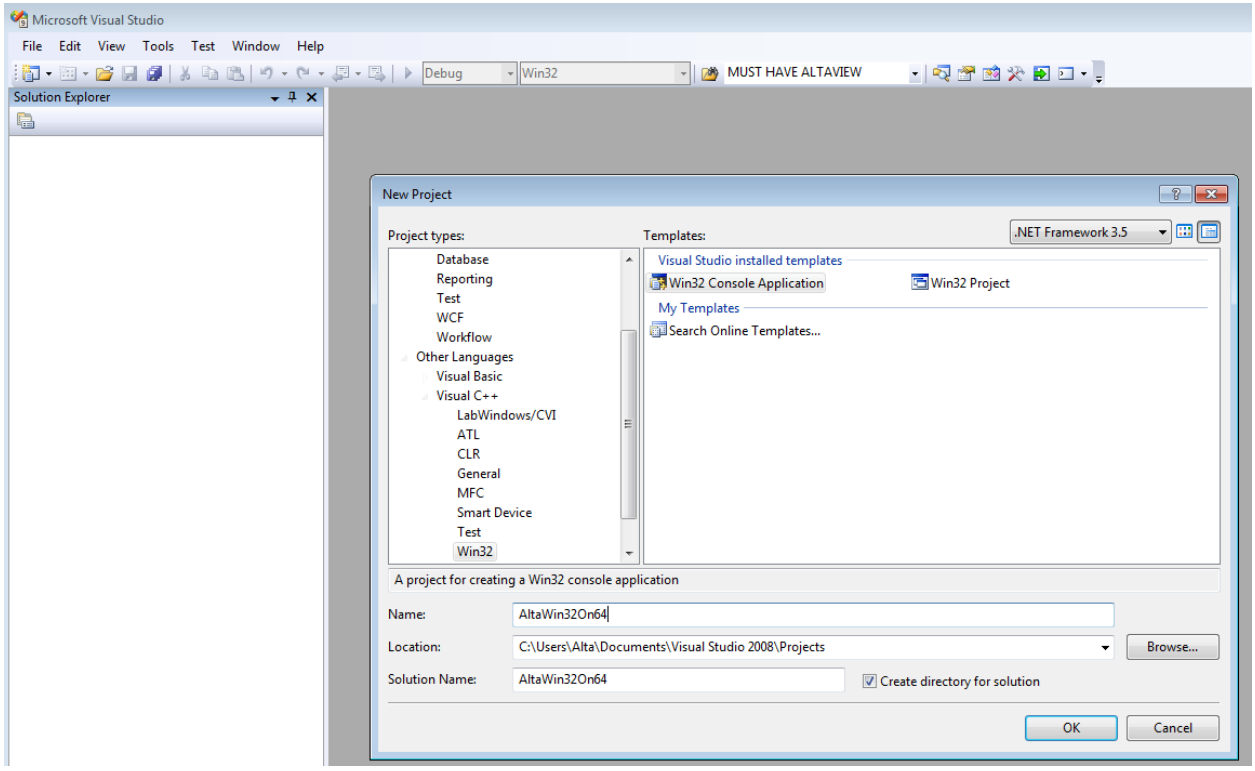
Whether you choose to build a 64-bit application for 64-bit OS or a 32-bit application for a 64-bit OS, use these Layer 0 and Layer 1 *header* files:
C:\Program Files\Alta Data Technologies\Alta Software\ADT_L0_API\include\ADT_L0.h
C:\Program Files\Alta Data Technologies\Alta Software\ADT_L1_API\include\ADT_L1.h

Running an existing 32-bit Alta application on 64-bit OS:
If you have a Win32 application that was built previously on a 32-bit system, that application *can* be executed on a 64-bit OS **by using the 32-bit DLLs from the 64-bit install instead of the 32-bit DLLs from the 32-bit install, i.e.:**
Use these DLLs:
C:\Program Files\Alta Data Technologies\Alta Software\ADT_L0_API**x86**\bin\ADT_L0.dll
C:\Program Files\Alta Data Technologies\Alta Software\ADT_L1_API**x86**\bin\ADT_L1.dll

For those users who are familiar with creating projects in MC Visual Studio, the above is all you need to know.

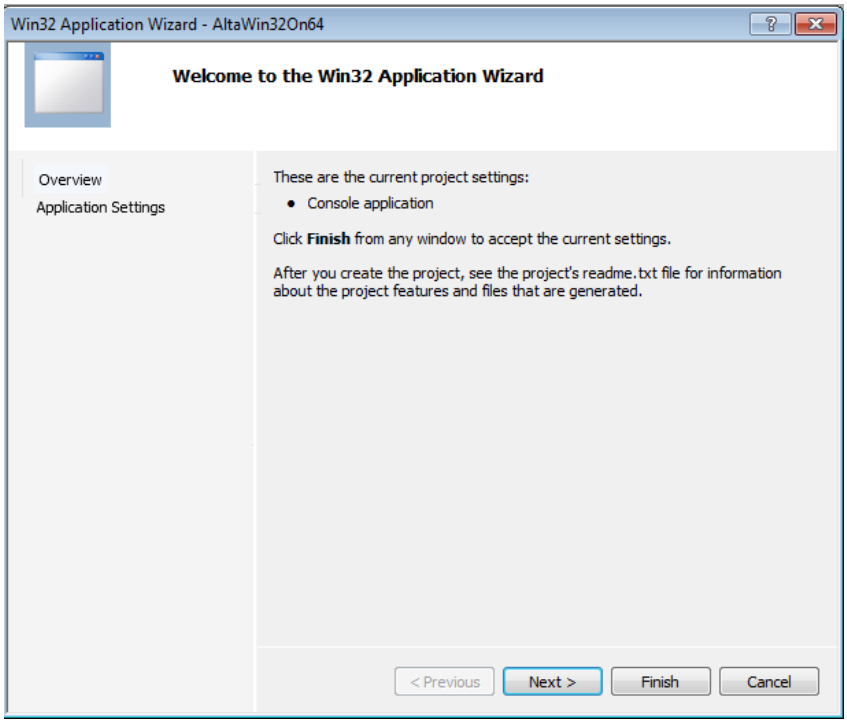
For the rest of us, we'll continue with the example build.

1) Create a Win32 project:

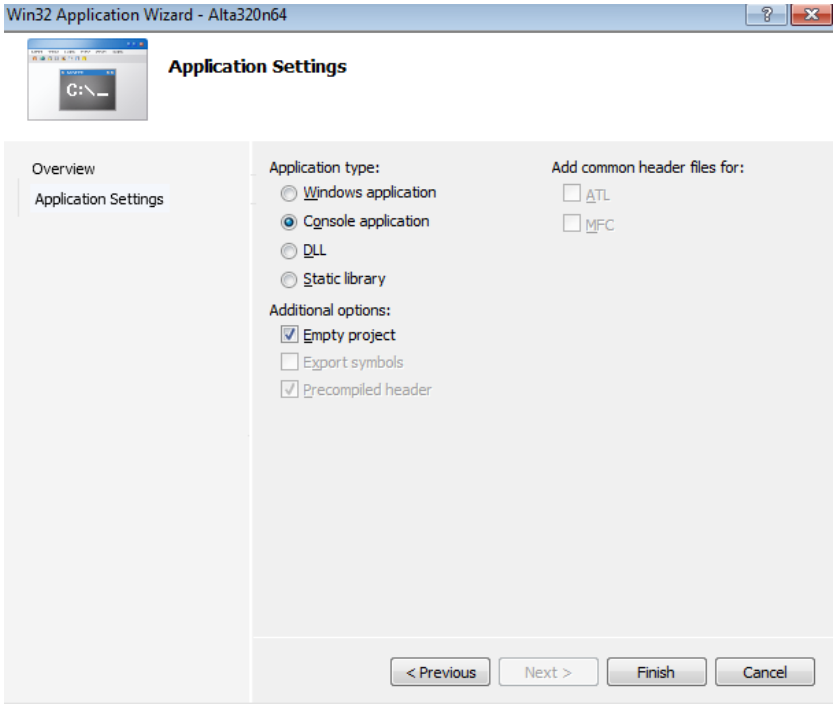


The project name chosen for this example is AltaWin32On64. Choose your own or use this, then click OK.

2) Select Next to continue to create empty project:



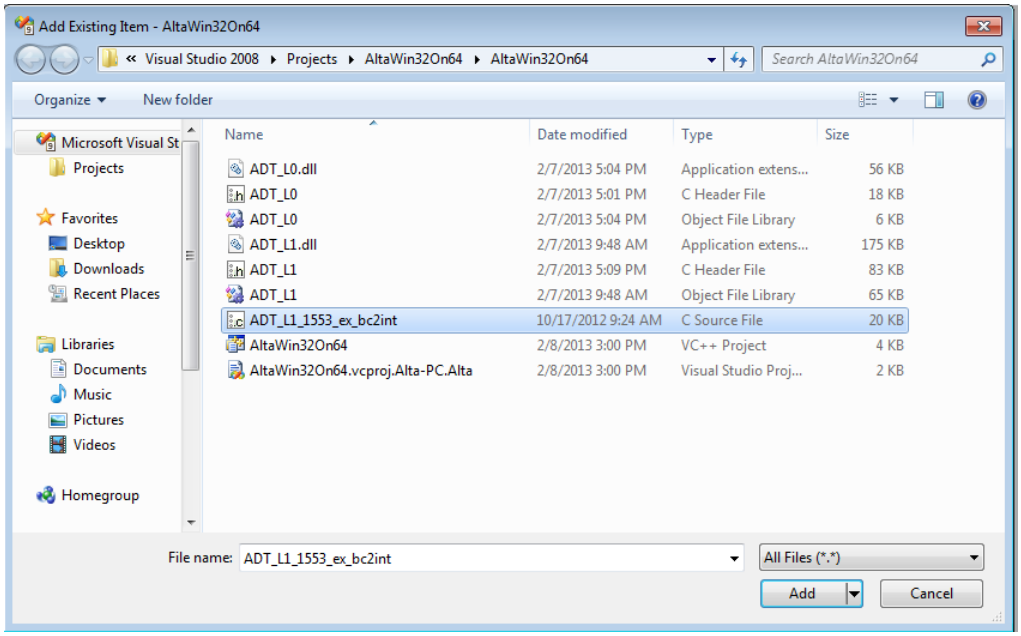
3) Ensure Console application is checked and select **Empty project**, then click **Finish.**



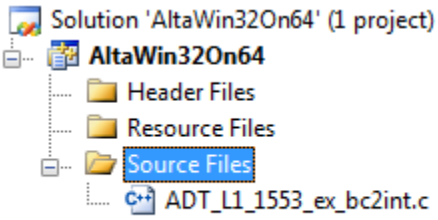
4) Open Windows Explorer and the folder **C:<yourProjectPath>\Visual Studio 2008\Projects\AltaWin32On64\AltaWin32On64**.

   Copy these files to the project folder above:
   C:\Program Files\Alta Data Technologies\Alta Software\ADT_L0_API**x86**\bin\ADT_L0.dll
   C:\Program Files\Alta Data Technologies\Alta Software\ADT_L0_API**x86**\bin\ADT_L0.lib
   C:\Program Files\Alta Data Technologies\Alta Software\ADT_L1_API**x86**\bin\ADT_L1.dll
   C:\Program Files\Alta Data Technologies\Alta Software\ADT_L1_API**x86**\bin\ADT_L1.lib
   C:\Program Files\Alta Data Technologies\Alta Software\ADT_L0_API\include\ADT_L0.h
   C:\Program Files\Alta Data Technologies\Alta Software\ADT_L1_API\include\ADT_L1.h
   C:\Program Files\Alta Data Technologies\Alta Software\ADT_L1_API\examples\M1553 Examples\ ADT_L1_1553_ex_bc2int.c

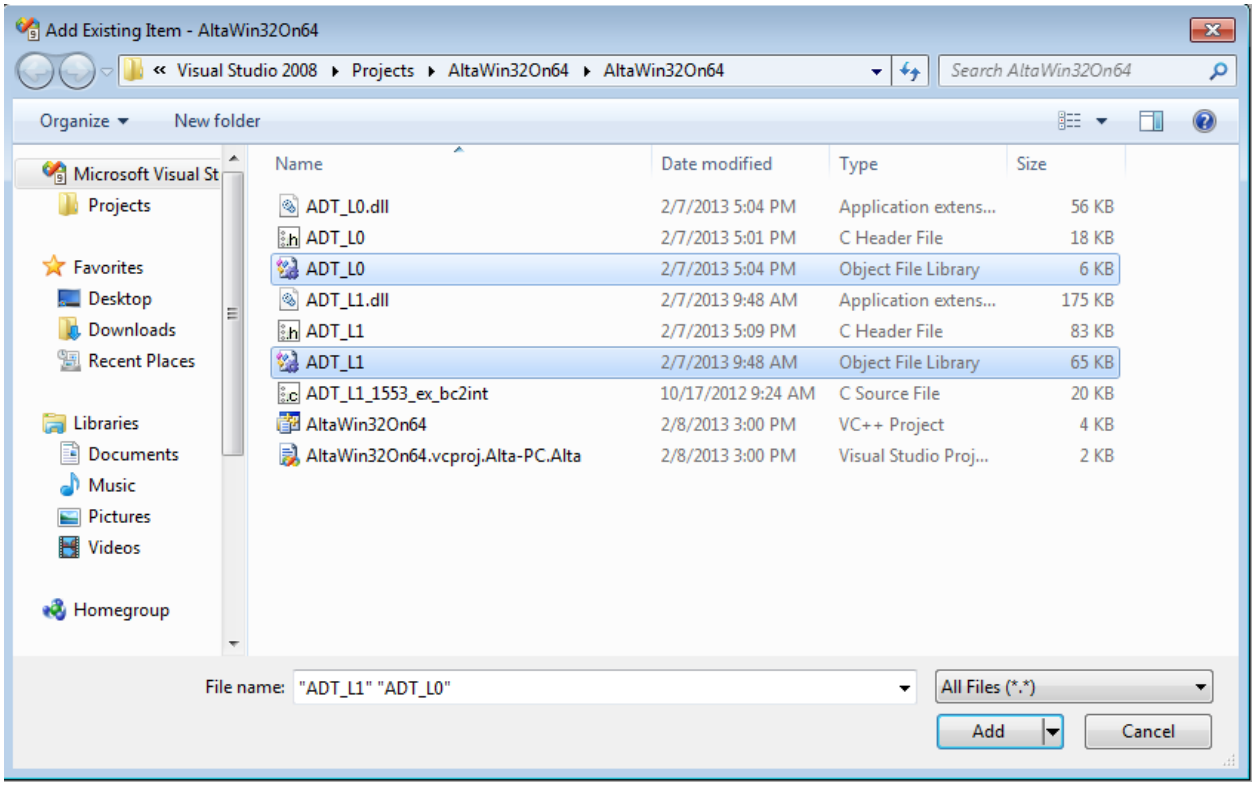
5) In Visual Studio, right-mouse-click on Source->Add Existing Item… to add the example source to your project. Select **ADT_L1_1553_ex_bc2int.c,** then click Add.



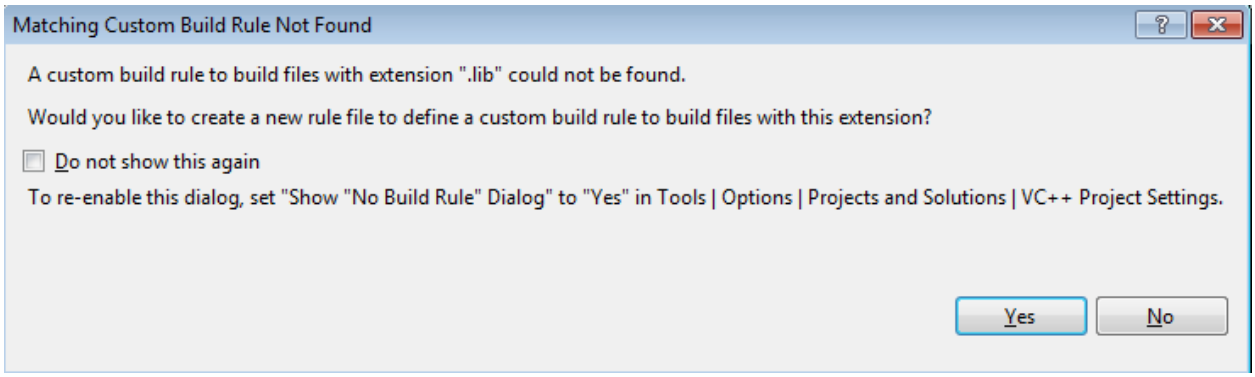
6) Your project should now look like this:



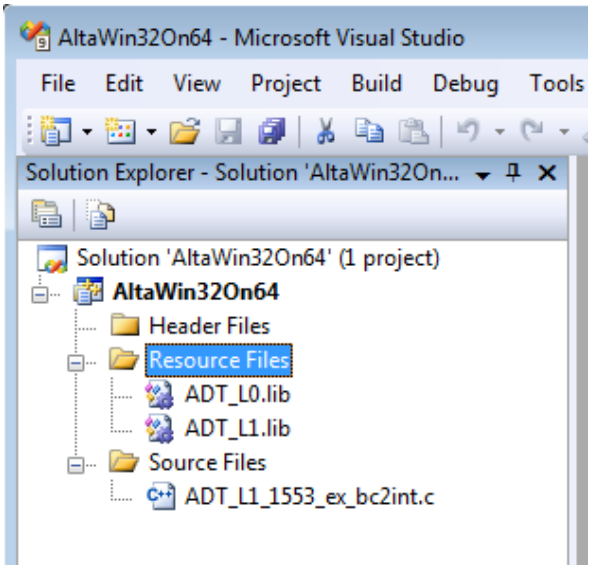
7) Right-mouse-click on Resource Files->Add Existing Item… to add the ADT_L0.lib and ADT_L1.lib files (click Add after selection of both):



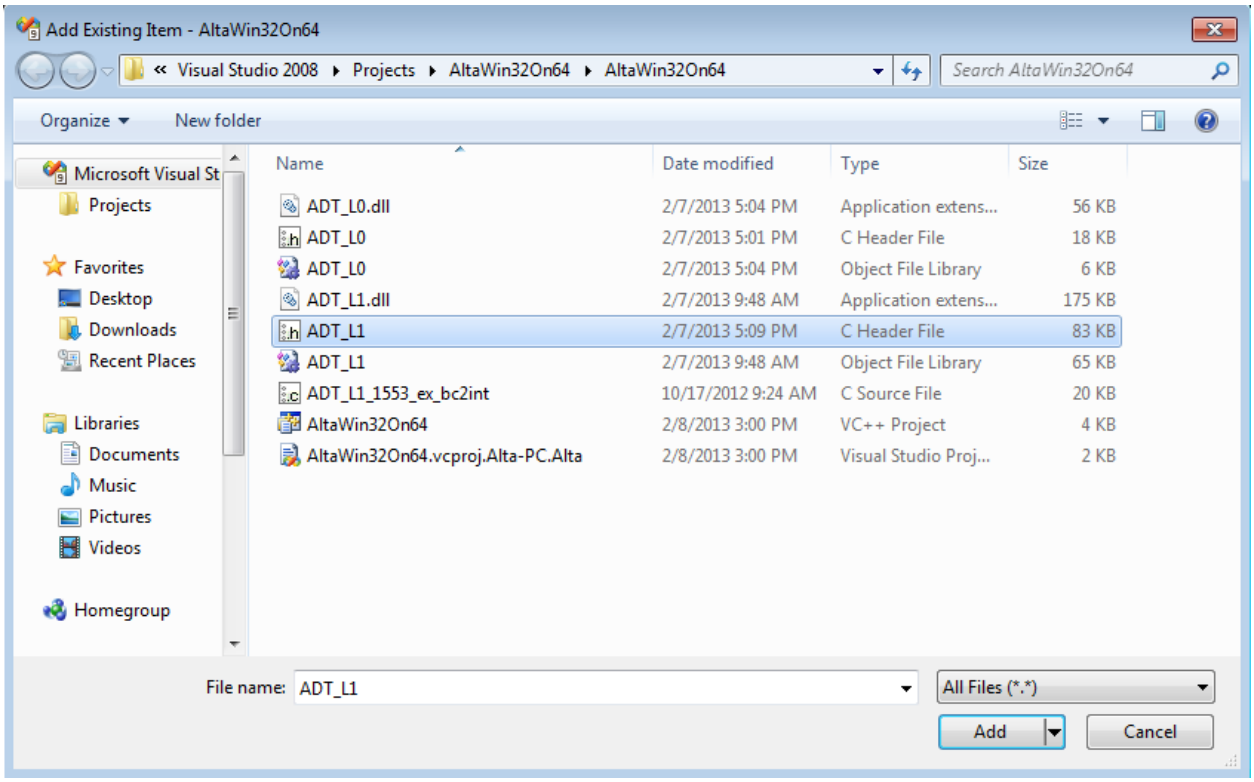
8) If this dialog box appears, click No:



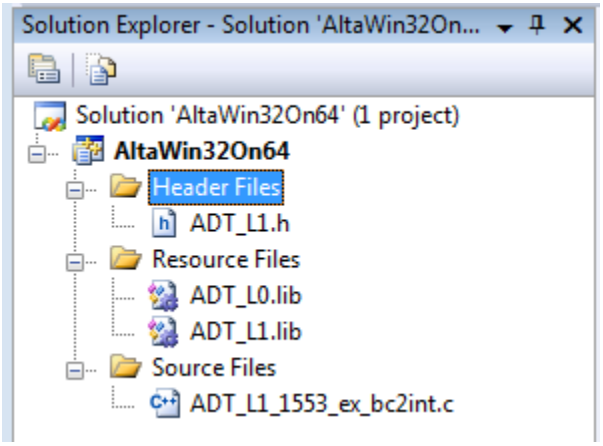
9) Your project should now look like this:



10) Right-mouse-click on Header Files->Add Existing Item… to add the ADT_L1.h header file:



11) Your project should now look like this:



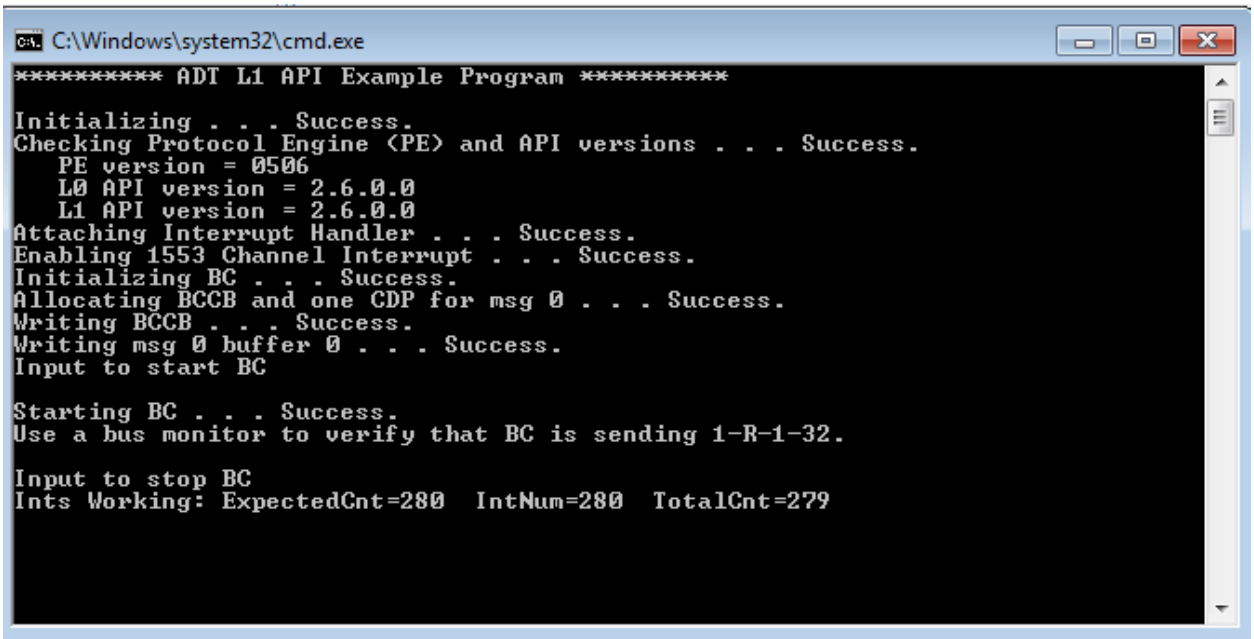
Open the ADT_L1_1553_ex_bc2int.c example source in the editor and ensure that the DEVID string matches the Alta hardware product installed. i.e.:

```
#define DEVID (ADT_PRODUCT_PMC1553 | ADT_DEVID_BOARDNUM_01 | ADT_DEVID_CHANNELTYPE_1553 | ADT_DEVID_CHANNELNUM_01).
```

For a description on defining the DEVID string for your product, refer to the Alta API Users Manual, in the section titled "**The Device Identifier (Device ID – or DEVID)".**

12) Build and run the project.

13) If build and run is successful, the Win32 console output for the 1553 example should look like:



14) This example output shows that the device was initialized properly, the BC was started (after user input to continue) and that interrupts are being generated and handled.