Using the Altia C/C++ API with Microsoft Developer Studio

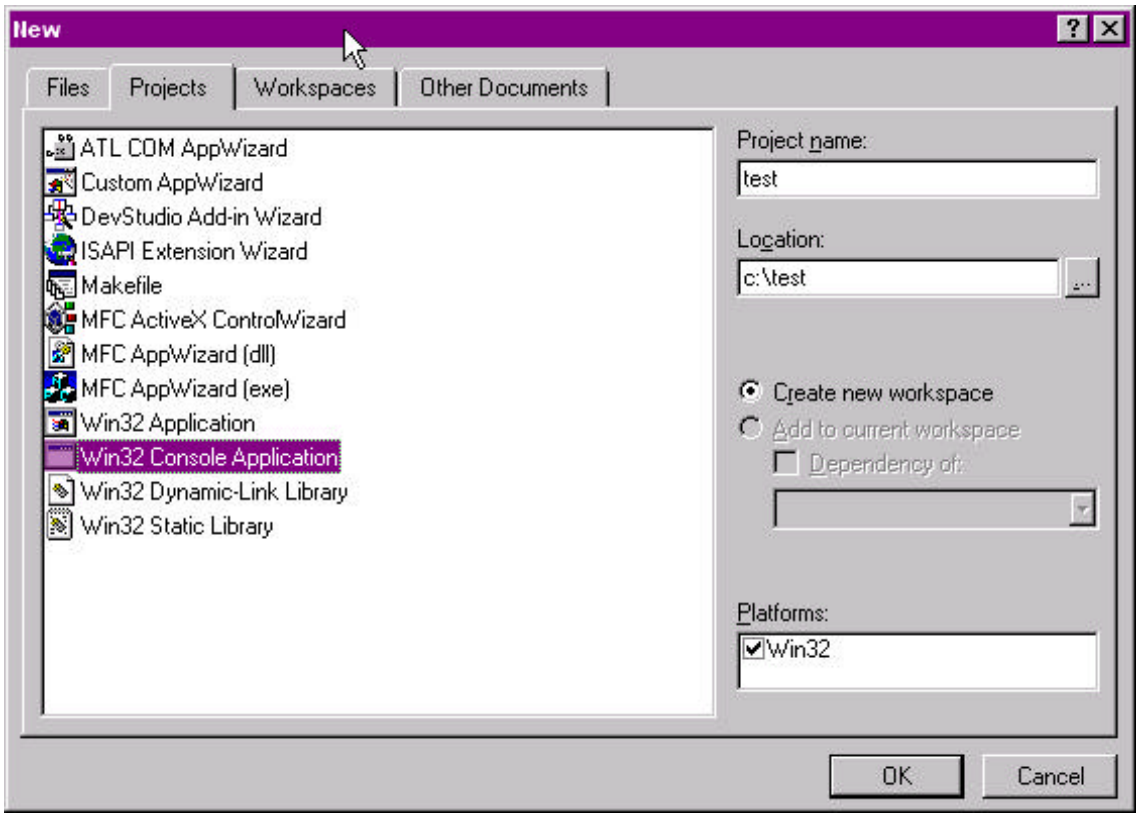
Introduction

Although the Altia C/C++ API comes with various example *makefiles* for use in compiling and linking C/C++ code that communicates with an Altia interface, many users prefer to use Microsoft's Integrated Development Environment (IDE), Developer Studio (MSDev). This Application Note details the MSDev settings that are necessary for building programs that can send and receive Altia data.

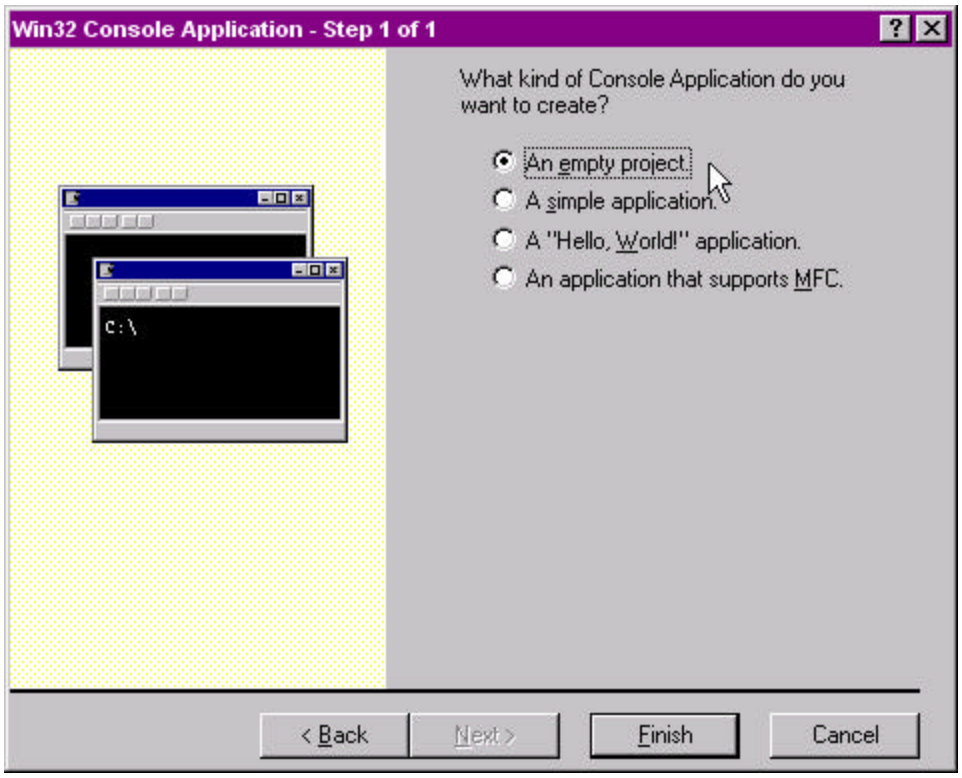
There are only two extra files that are required to take a boring bit of C code and make it Altia enabled: `altia.h` and `libdde.lib`. We'll discuss how to include these two files into your MSDev project, but first let's discuss how to create a new project.

Creating a New Project in MSDev

Open MSDev and choose **New** from the **File** menu. In the dialog that opens, choose **Win32 Console Application** and give your project a name.



In some later versions of MSDev, a second dialog will open after you press **OK**. If it does, just choose to create **An empty project** (as shown below) and continue on.



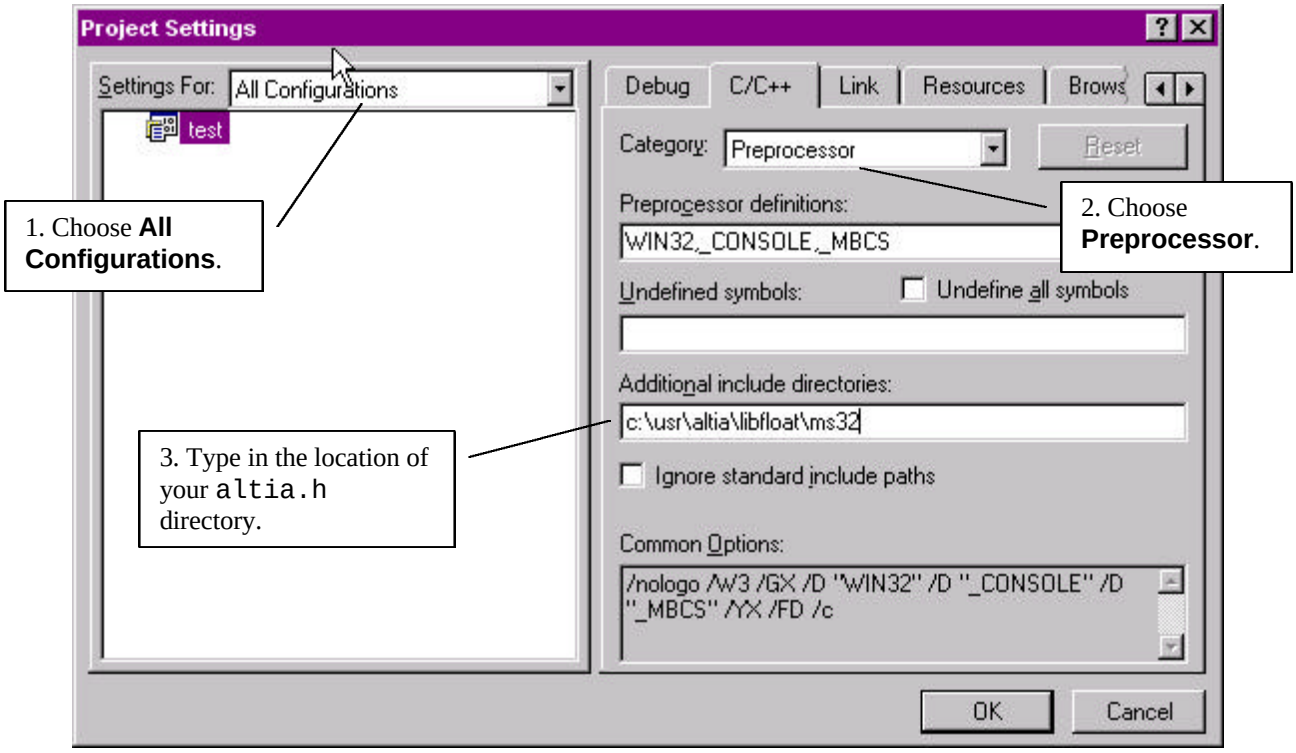
Now, you are ready to develop your code. Read on to find out how to add the all-important Altia header and library files to your project.

Adding the Altia Header File

Really, the inclusion of the Altia header file `altia.h` is done via a `#include` statement in your code. This section discusses how to make sure that your project can find the right version of that file.

From the **Project** menu, choose the **Settings…** option. In the dialog that opens, change to the **C/C++** tab and do the following:

1. Change the **Settings For:** drop down box to **All Configurations**.
2. Change the **Category:** drop down box to **Preprocessor**.
3. In the **Additional include directories:** field, type in the directory where your Altia header file is located (e.g., **`c:\usr\altia\libfloat\ms32`**).
4. Press the **OK** button .

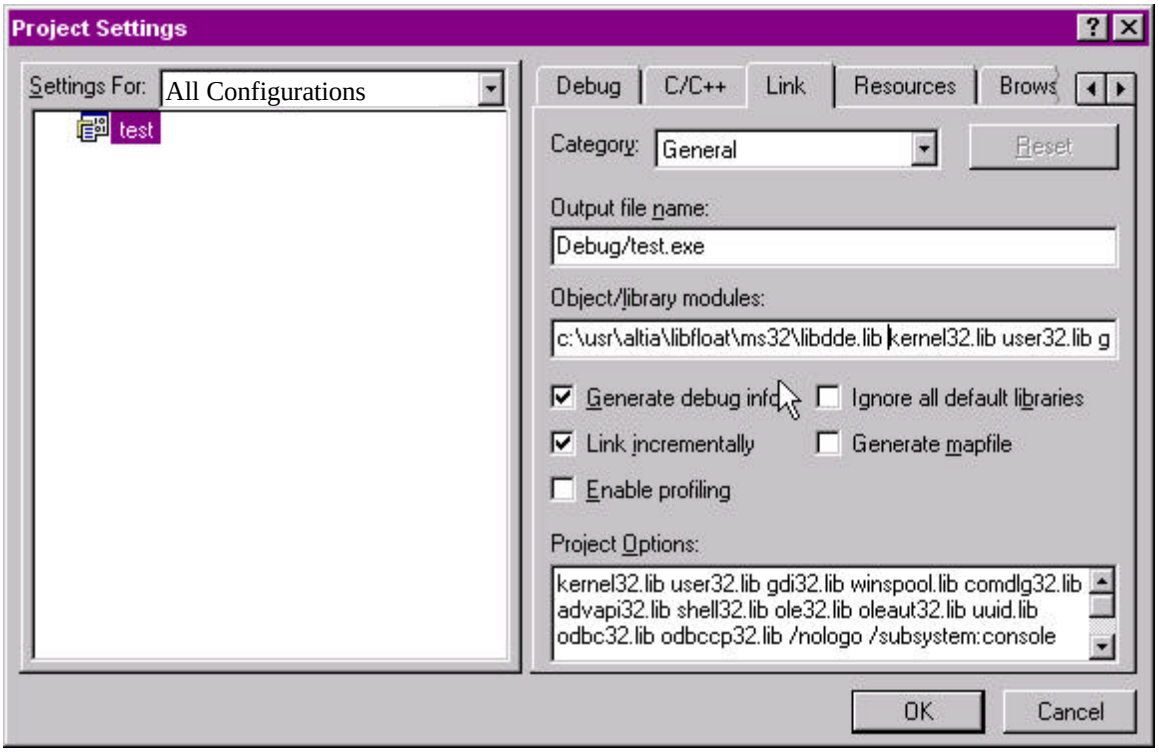


That's it. Your project now knows where to look when it sees a `#include <altia.h>` line in your source code.

Adding the Altia Library File

To add the `libdde.lib` file to your project, once again choose **Settings…** from the **Project** menu. As with adding the Altia header file, be sure that the **Settings For:** drop down box is displaying the **All Configurations** option.

This time, go to the **Link** tab of the dialog. Then, in the **Object/library modules:** field, enter the full path to the Altia API `libdde.lib` library (e.g., `c:\usr\altia\libfloat\ms32\libdde.lib`) **before** all other existing object/library modules. Press the **OK** button after making this change.

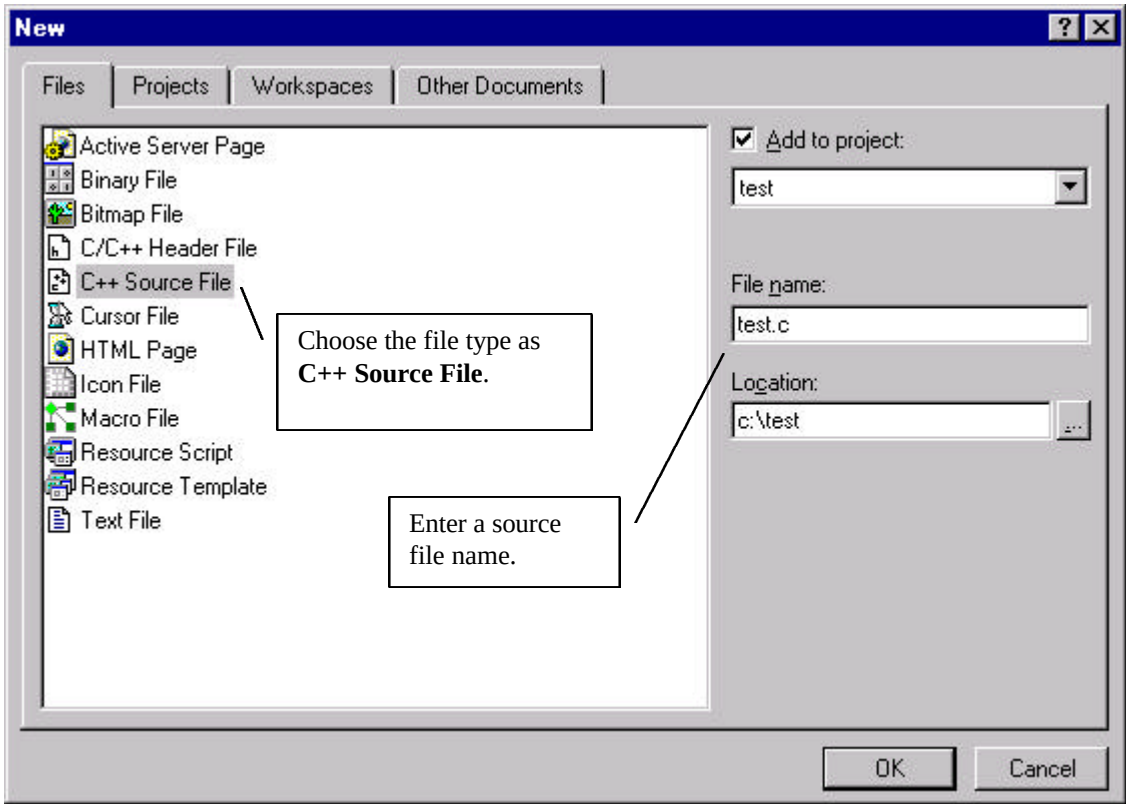


All done! Now you are ready to add source code to your project, build it into an executable program, and finally execute the program.

Adding a Source File

Your project must have C or C++ source code that defines the program logic. For example, you might want to have a file named `test.c` in your project containing C source code.

From the **File** menu, choose the **New…** option. In the New dialog, select to add a C++ Source File and enter the name of the file as demonstrated in the dialog below. Finish by pressing the **OK** button.



When this dialog closes, MSDev will display the new source file (e.g., `test.c`) in its text editor area. Of course, the file is empty and you need to type in source code for at least a `main()` routine. For example, you could type in something as simple as the code shown below:

```
#include <stdio.h>
#include <altia.h>

int main (int argc, char *argv[ ])
{
      AtConnectId altia_id;

      /* Try to connect to an Altia session already running */
      if ((altia_id = AtOpenConnection(NULL,NULL,argc,argv)) < 0)
      {
            printf("AtOpenConnection failed – no Altia!\n");
            return 1;
      }
```

```
        /* We must've connected.  Send events to Altia. */
        AtSendEvent(altia_id, "move", (AltiaEventType) 0);
        AtSendEvent(altia_id, "move", (AltiaEventType) 100);

        /* Close the connection.  An important step! */
        AtCloseConnection(altia_id);

        return 0;
}
```

Building a Project into a Program

After you type in a sufficient amount of source code (such as the code displayed above), you are ready to build your program by choosing **Build project_name.exe** or **Rebuild All** from the **Build** menu.

After a successful build, the status window at the bottom of MSDev should display something like:

```
Compiling...
test.c
Linking...
LINK : warning LNK4098: defaultlib "LIBC" conflicts with use
of other libs; use /    NODEFAULTLIB:library

test.exe - 0 error(s), 1 warning(s)
```

These are very good results! You can ignore the `LINK: warning LNK4098:` message. If you see something other than `0 error(s)`on the last line, you will need to carefully view the previous lines in the status window to determine what went wrong. Normally, you can double-click on a specific error or warning line in the status window and it will select the line in your source code that is associated with the problem.

Executing a Program

After your program builds successfully, execute it by choosing **Execute project_name.exe** from the **Build** menu. Your program is a Win32 Console Application so it will display a Console window as it executes. This is extremely helpful for displaying debug messages using the `printf()` function as shown in the example.

You can stop your program (some might call it exiting or ending the program) at any time by closing the Console window via its **X** button in the upper right corner. You can also stop the program by pressing Ctrl+C when the Console window is the active window.

Please note that you cannot successfully rebuild your project while your program is running. This is because MSDev cannot overwrite your executable program if it is running. If you try to rebuild your project while your program is running, you will probably see errors in the status window that look like:

```
LINK : fatal error LNK1168: cannot open test.exe for writing
Error executing link.exe.

test.exe - 1 error(s), 0 warning(s)
```

The solution, of course, is to stop your program and then rebuild your project.

You have made a completely self-sufficient program at this point that you can execute outside of the MSDev environment. Go ahead and use an Explorer or My Computer window to browse to the folder containing your program (e.g., `c:\test\Debug`) and execute the program (e.g., `test.exe`) by double-clicking on it. It will open in a Console window and the window will automatically close if the program runs to completion. You can move your `.exe` program to any folder or any Windows 95/98/NT/2000 machine and it will execute in a similar fashion.

If you would like to start Altia Runtime with your `.dsn` design file and start the program in one simple step, create a folder that contains the following files:

`go.bat`	- A script that you create (see explanation below)
`project.dsn`	- Your Altia `.dsn` design file
`project.rtm`	- Altia Runtime configuration file associated with your `.dsn` design file
`project.exe`	- Your program from your MSDev project folder (e.g., `c:\test\Debug`)
`altiart.exe`	- Altia Runtime from Altia software `bin` folder (e.g., `c:\usr\altia\bin`)
`fonts.ali`	- Altia Runtime support file also from Altia software `bin` folder
`colors.ali`	- Altia Runtime support file also from Altia software `bin` folder

The `go.bat` script is something that you would create using a simple text editor such as Notepad. It would typically contain a single line that looks like:

```
altiart.exe  –file  project.dsn –exec  project.exe
```

This relatively simple command will start Altia Runtime. Altia Runtime will open `project.dsn` using the configuration information provided by `project.rtm` and finally execute `project.exe` after `project.dsn` is fully loaded and initialized.

A user can run the complete project by simply double-clicking on `go.bat`.

As an alternative, your C/C++ source code can start Altia Runtime directly by executing `AtStartInterface()` in place of `AtOpenConnection()`. In this case, a `go.bat` or similar script is unnecessary. Instead, a user would simply double-click on `project.exe` to run the complete project. See the Altia API Reference Manual for a detailed description of `AtStartInterface()` and how to use it.

Please send questions or comments regarding this Application Note to support@altia.com or visit us on the web at www.altia.com.