

Getting Started with MASM

Updated 8/26/2008, for Visual Studio 2008

If you've recently purchased **Assembly Language for Intel-Based Computers**, 5th edition, you probably want to get the software set up so you can start working. This tutorial should make the process easier. If you're in a hurry to get started, you only need to read Item 1.

1. [Required setup for 32-bit applications](#)
2. [Building 16-bit applications](#) (Chapters 12-16)
3. [Project properties settings](#)
4. [Creating a project from scratch](#)
5. [Generating a source listing file](#)
6. [Using the Visual Studio 2005 debugger](#)
7. [MASM syntax highlighting](#)
8. [Assembling, linking, and debugging with a batch file](#)

Found an error in this document? Please [email me immediately](#). Except where noted, all instructions in this document apply equally to Visual Studio 2008 and Visual C++ 2008 Express.

Required Setup for 32-bit Applications

If you are using **Visual Studio 2008 Professional or Team Suite**, these products already contain the Microsoft Assembler 8.0. You can skip Steps 1 and 2 and [go directly to Step 3](#).

Step 1: Install Visual C++ 2008 Express Edition, Service Pack 1

You can download [Visual C++ 2008 Express \(SP1\)](#) from Microsoft.

When you run the Visual C++ setup program, make a note of the location where the C++ compiler is installed. This information will be useful to you later.

From now on, we will leave the "2008" out of the name *Visual C++ Express*.

Step 2: Download and Install the Microsoft Assembler

Visit Microsoft's MASM 8.0 download site.

Follow the download and installation instructions on the Microsoft page. If the link is broken, please [let us know by email](#).

Step 3: Installing the Book's Example Programs

[Click this link](#) to get the latest copy of the book's link libraries and example programs. The examples are stored in a self-extracting archive file that automatically extracts to the **c:\Irvine** folder. Unless you have some objection to using that location, do not alter the path. (Lab managers: you can designate c:\Irvine directory as read-only.) If you plan to change the installation location, read our instructions relating to [changing project properties](#).

The following files will be copied into the c:\Irvine directory:

Filename	Description
----------	-------------

GraphWin.inc	Include file for writing Windows applications
--------------	---

Irvine16.inc	Include file used with the Irvine16 link library (16-bit applications)
--------------	--

Irvine16.lib	16-bit link function library used with this book
--------------	--

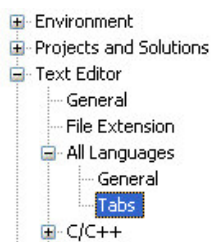
Irvine32.inc	Include file used with the Irvine32 link library (32-bit applications)
Link16.exe	16-bit linker
Irvine32.lib	32-bit link function library used with this book
Macros.inc	Include file containing macros (explained in Chapter 10)
SmallWin.inc	Small-sized include file, used by Irvine32.inc
make16.bat	Batch file for building 16-bit applications
VirtualKeys.inc	Keyboard code definitions file, used by Irvine32.inc

A subdirectory named **Examples** will contain all the example programs shown in the book.

Step 4: Building a Sample Assembly Language Program

Preliminary Step: Set Tab Size to 5

Start Visual C++ Express, and select **Options** from the **Tools** menu. Select **Text Editor**, Select **All Languages**, and select **Tabs**:



Set the Tab Size and Indent Size to 5.

Opening a Project

Visual Studio and Visual C++ Express require assembly language source files to belong to a *project*, which is a kind of container. A **project** holds configuration information such as the locations of the assembler, linker, and required libraries. A project has its own folder, and it holds the names and locations of all files belonging to it. We have created a sample project folder in the **c:\Irvine\Examples** directory, and its name is **Project_Sample**.

Do the following steps, in order:

1. Start Visual Studio or Visual C++ Express.
2. If you're using Visual Studio, select **Open Project** from the File menu. Or, if you're using Visual C++ Express, select **Open**, and select **Project/Solution**.
3. Navigate to the **c:\Irvine\Examples\Project_Sample** folder and open the file named **Project.sln**.
4. In the *Solution Explorer* window, click the + symbol next to the item named **Project** to expand it. Double-click the file named **main.asm** to open it in the editing window. (*Visual Studio users may see a popup dialog asking for the encoding method used in the asm file. just click the OK button to continue.*)

Tip: If the *Solution Explorer* window is not visible, select **Solution Explorer** from the *View* menu. Also, if you do not see *main.asm* in the *Solution Explorer* window, look at the tabs along the bottom of the window. Click the **Solution Explorer** tab.

You should see the following program in the editor window:

```
TITLE MASM Template                                (main.asm)

; Description:
;
; Revision date:

INCLUDE Irvine32.inc

.data
myMessage BYTE "MASM program example",0dh,0ah,0

.code
main PROC
    call Clrscr

    mov  edx,OFFSET myMessage
    call WriteString

    exit
main ENDP

END main
```

Later, we'll show you how to copy this program and use it as a starting point to write your own programs.

Build the Program

Next, you will build (assemble and link) the sample program:

- If you're using Visual C++ Express, select **Build Solution** from the **Build** menu.
- If you're using Visual Studio, select **Build Project** from the **Build** menu.

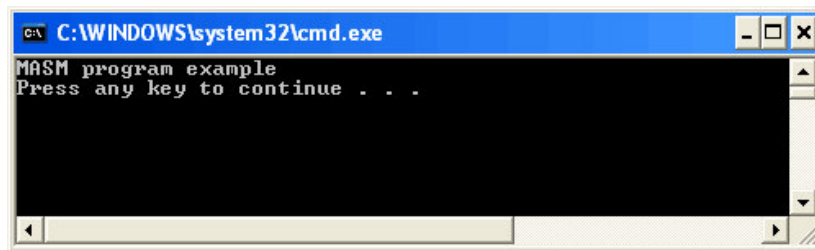
In the output window at the bottom of the screen, you should see messages similar to the following, indicating the build progress:

```
1>----- Build started: Project: Project, Configuration: Debug Win32 -----
1>Assembling...
1>Assembling: .\main.asm
1>Linking...
1>Embedding manifest...
1>Build log was saved at "file:///g:/masm/Project_sample/Debug/BuildLog.htm"
1>Project - 0 error(s), 0 warning(s)
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

If you do not see these messages, the project has probably not been modified since it was last built. No problem--just add a space somewhere in the document, save it, and try the Build command again.

Run the Program

Select **Start without Debugging** from the **Debug** menu. The following console window should appear, although your window will be larger than the one shown here:



The "Press any key to continue..." message is automatically generated by Visual C++ Express.

Congratulations, you have just run your first Assembly Language program.

Press any key to close the Console window.

When you assembled and linked the project, a file named **Project.exe** was created inside the project's \Debug folder. This is the file that executes when you run the project. You can execute Project.exe by double-clicking its name inside Windows Explorer, but it will just flash on the screen and disappear. That is because Windows Explorer does not pause the display before closing the command window.

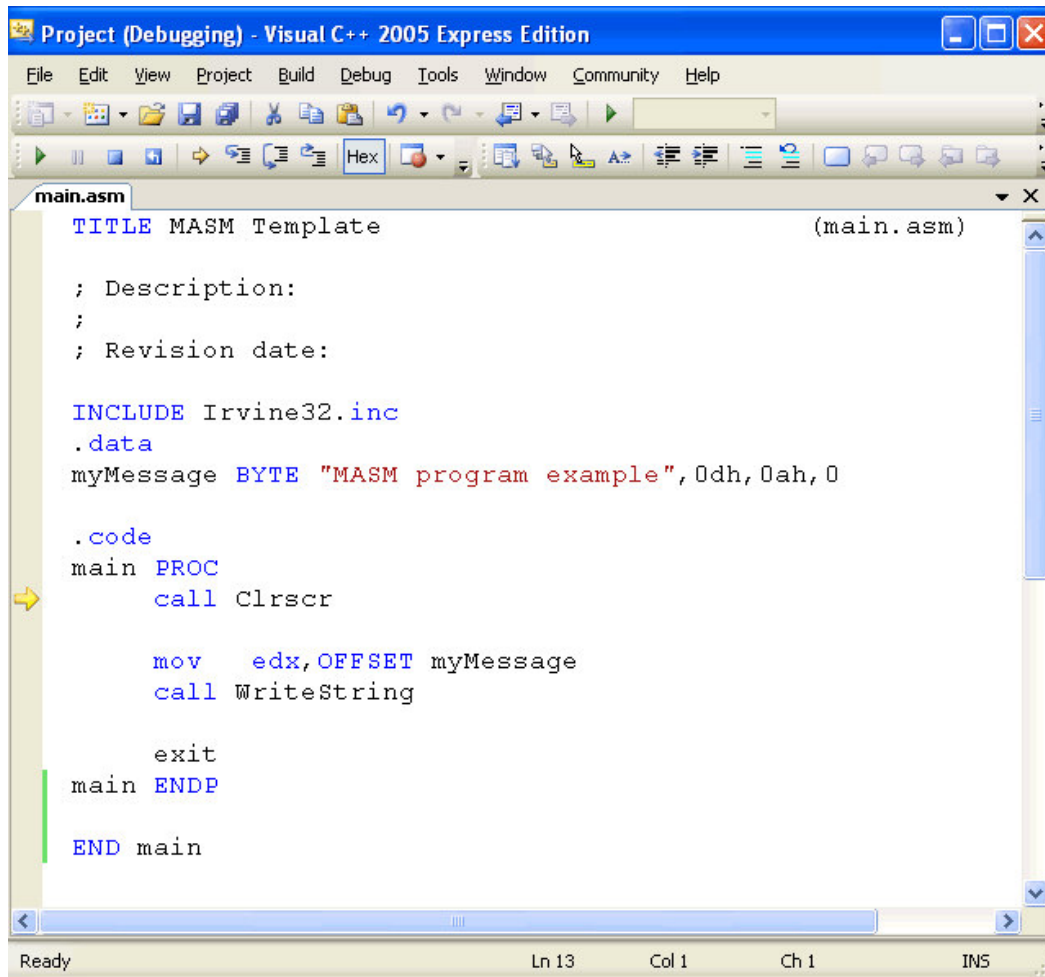
Creating New Projects of Your Own

Before long, you will want to create your own projects. The easiest way to do this is to copy the entire **c:\Irvine\Examples\Project_Sample** folder to a new location. Copy it to a folder in which you have read/write permissions. (If you're working in a college computer lab, a useful location is a portable USB drive. Then you can modify the program, build, and run it again.

Step 5: Running the Sample Program in Debug Mode

In this step, you will set a breakpoint inside the sample program. Then you will use the Visual C++ debugger to step through the program's execution one statement at a time.

1. To begin stepping through your program in Debug mode, press the F10 key.
2. A yellow arrow will appear next to the first program statement (*call Ckrscr*). The arrow indicates that the statement is next to be executed.
3. Press the F10 key (called *Step Over*) to execute the current statement. Continue pressing F10 until the program is about to execute the **exit** statement.
4. A small black window icon should appear on your Windows status bar. Open it and look at the contents of the Command window. You should see the words "MASM program example" in the window.
5. Press F10 one more time to end the program.



Registers

If you want to display the CPU registers, do the following: Start debugging the program, then select *Windows* from the *Debug* menu. Select *Registers* from the drop-down list. The bottom window will display the register contents. Right click this window and check the item *Flags* to enable the display of conditional flags.

You can interrupt a debugging session at any time by selecting *Stop Debugging* from the *Debug* menu. You can do the same by clicking the blue square button on the toolbar. To remove a breakpoint from the program, click on the red dot so that it disappears.

Setting a BreakPoint

If you set a breakpoint in a program, you can use the debugger to execute the program a full speed (more or less) until it reaches the breakpoint. At that point, the debugger drops into single-step mode.

1. Click the mouse along the border to the left of the call `WriteString` statement. A large red dot should appear in the margin.
2. Select *Start Debugging* from the *Debug* menu. The program should run, and pause on the line with the breakpoint, showing the same Yellow arrow as before.
3. Press F10 until the program finishes.

You can remove a breakpoint by clicking its red dot with the mouse. Take a few minutes to experiment with the *Debug* menu commands. Set more breakpoints and run the program again. For the time being, you can use the F11 key to step through the program in the same way the F10 key did.

Building and Running Other Programs

Suppose you want to run another example program, or possibly create your own program. You can either edit and modify `main.asm`, or you can remove `main.asm` from the project and insert some other

.asm file into the project.

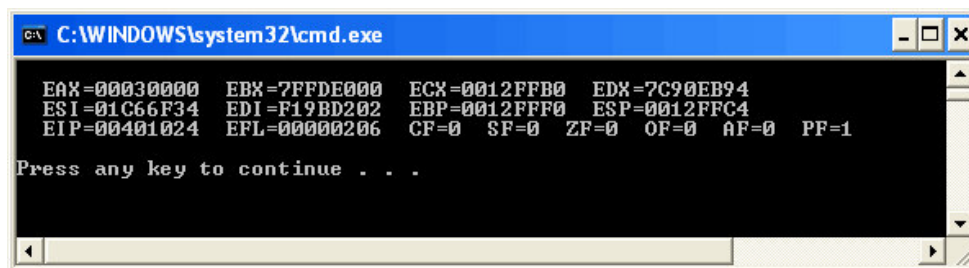
- To remove a program from a project without deleting the file, right-click its name in the *Solution Explorer window*. In the context menu, select **Exclude from Project**. If you change your mind and decide to add it back to the project, right-click in the same window, select **Add**, select **Existing item**, and select the file you want to add.
- To remove a program from a project and delete the source code file, select the file with the mouse and press the **Del** key. Or, you can right-click the file name and select **Remove**.

Adding a File to a Project

The easiest way to add an assembly language source file to an open project is to drag its filename with the mouse from a Windows Explorer window onto the name of your project in the Solution Explorer window. A reference to the file (not a copy) will be inserted in your project's directory. Try this now:

1. Remove the main.asm file from your project.
2. Add a reference to the file c:\Irvine\Examples\ch03\AddSub.asm to the project.
3. Build and run the project.

Here is what you should see in the Console window, except that only your EAX register will have the same value as ours:



```
C:\WINDOWS\system32\cmd.exe

EAX=00030000  EBX=7FFDE000  ECX=0012FFB0  EDX=7C90EB94
ESI=01C66F34  EDI=F19BD202  EBP=0012FFF0  ESP=0012FFC4
EIP=00401024  EFL=00000206  CF=0  SF=0  ZF=0  OF=0  AF=0  PF=1

Press any key to continue . . .
```

When you press a key, the console window will close.

Copying a source file

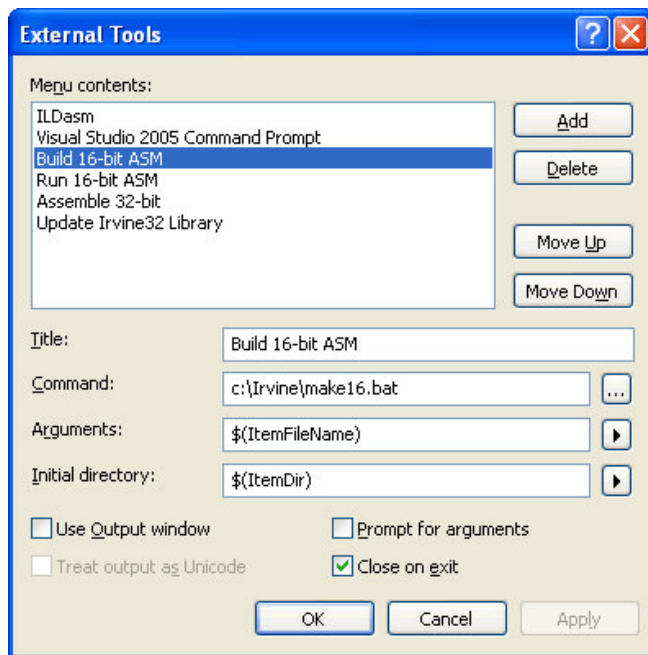
If you want to make a copy of an existing file, use Windows Explorer to copy the file into your project directory. Then, right-click the project name in Solution Explorer, select Add, select Existing Item, and select the filename.

[Return to top](#) or read about [Project Properties settings](#).

Building 16-bit Applications (Chapters 12-16)

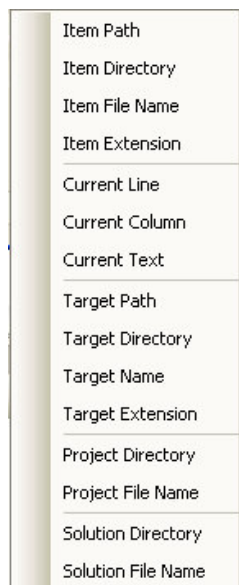
Only Chapters 12 through 16 require the building of 16-bit applications. Except for a few exceptions, which are noted in the book, your 16-bit applications will run under Windows XP and Windows Vista.

If you plan to build 16-bit applications, you need to add two new commands to the Tools menu in Visual C++ Express (or Visual Studio). To add a command, select **External Tools** from the Tools menu. The following dialog will appear, although many of the items in your list on the left side will be missing:



Step 1: Create the Build 16-bit ASM Command

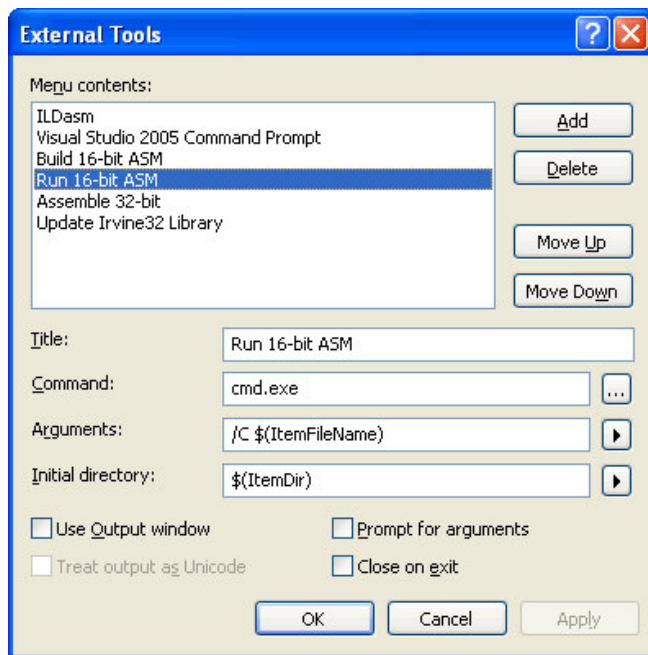
Click the **Add** button and fill in the Title, Command, Arguments, and Initial directory fields as shown in the screen snapshot. If you click the buttons with arrows on the right side of the Arguments and Initial directory fields, a convenient list appears. You can select an item without having to worry about spelling:



Click the **Apply** button to save the command.

Step 2: Create the Run 16-bit ASM Command

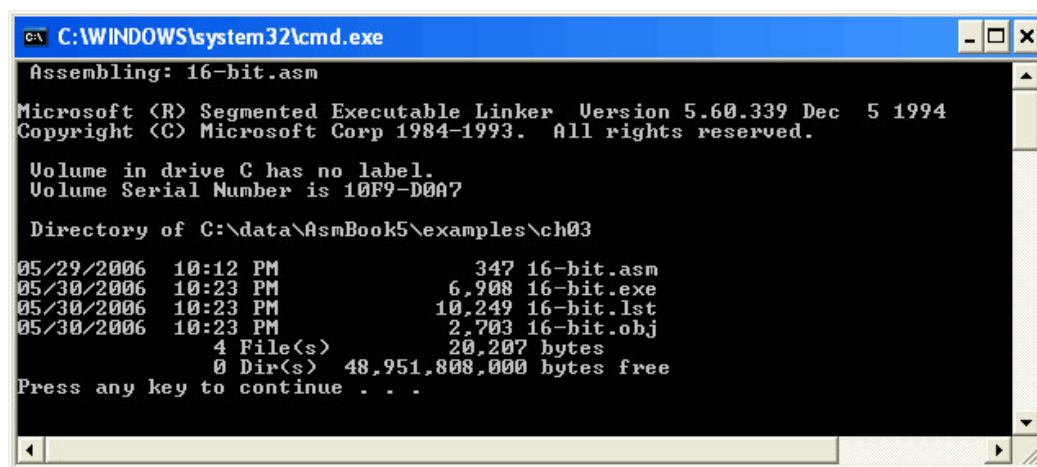
Click the Add button again, and create a new command named **Run 16-bit ASM**:



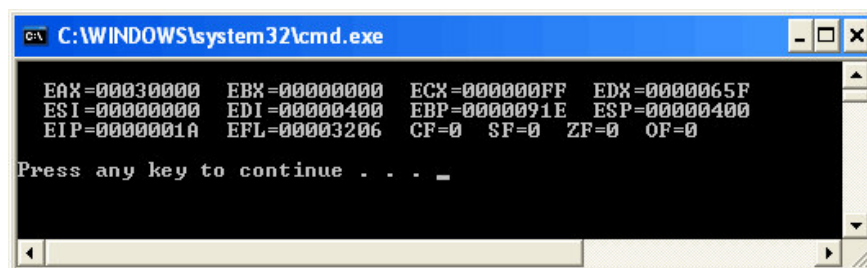
Click the OK button to save the command and close the External Tools dialog.

Testing Your new 16-Bit Commands

To test your new 16-bit commands, open the file named **16-bit.asm** from the ch03 folder in the book's example programs. Select **Build 16-bit ASM** from the Tools menu. The following command window should appear, showing the successful execution of the assembler and linker, followed by a listing of all files related to this program:



Press a key to close the window. Next, you will run the program. Select **Run 16-bit ASM** from the Tools menu. The following window will appear, although the contents of all registers except EAX will be different:



Press a key to close the window.

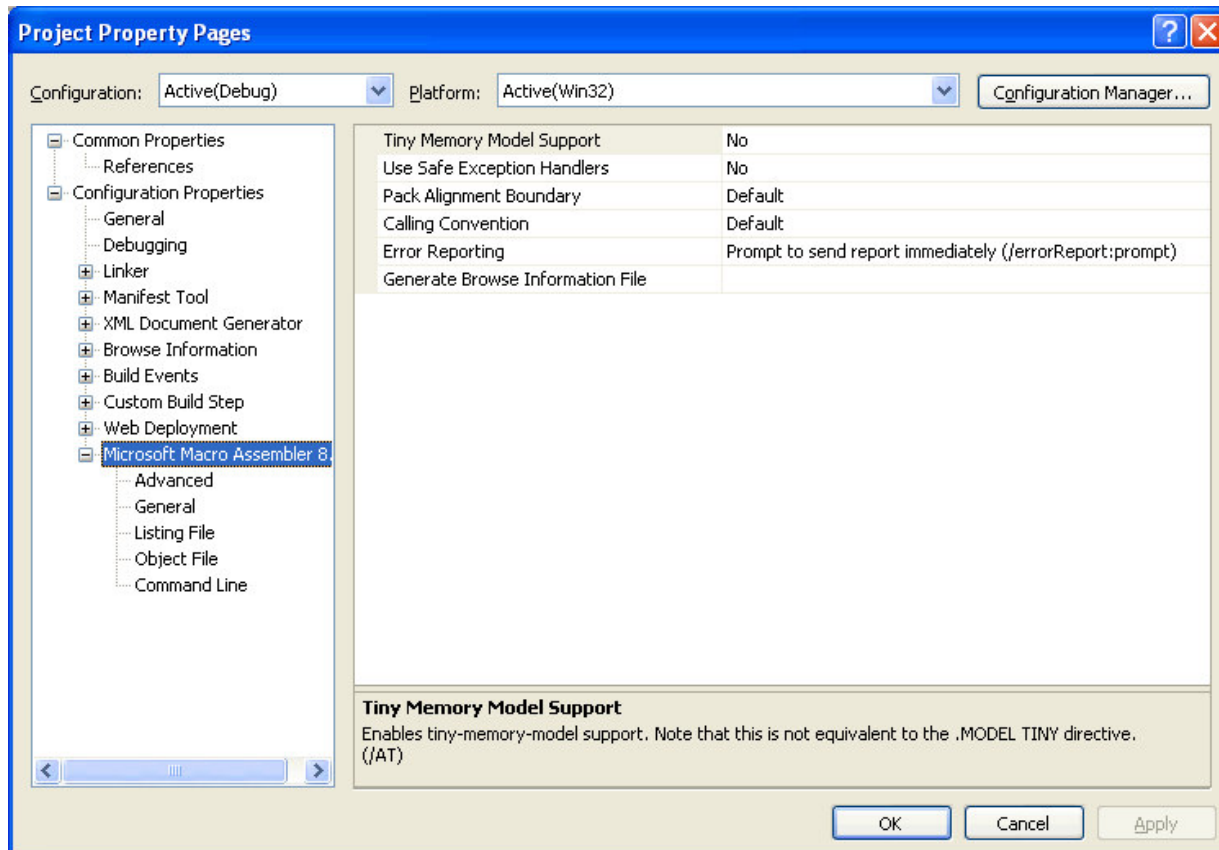
You have completed the setup for building and running 16-bit assembly language programs.

[Return to top](#)

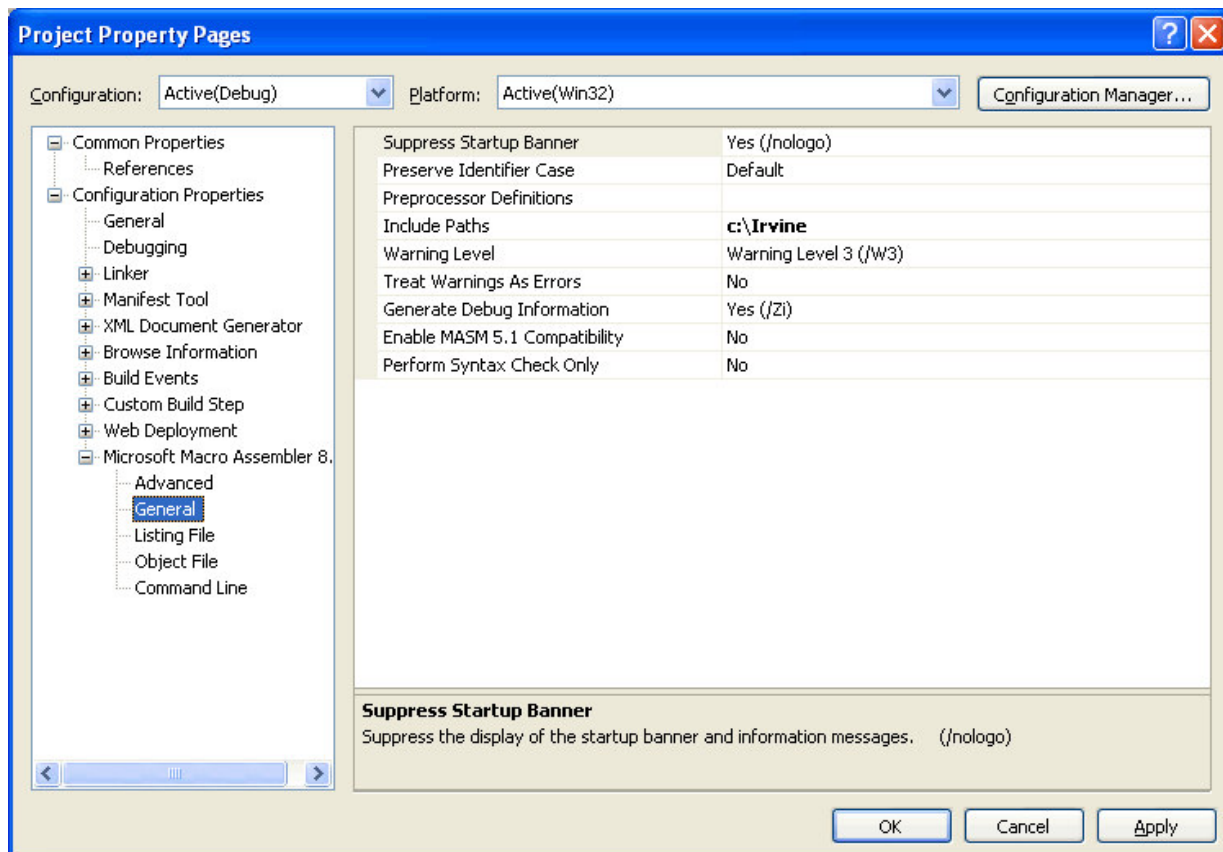
Project Properties Settings (Optional Topic)

We thought you might like to know more about how Visual C++ projects are set up for assembly language programs. If so, we will walk you through the important parts of the project configuration.

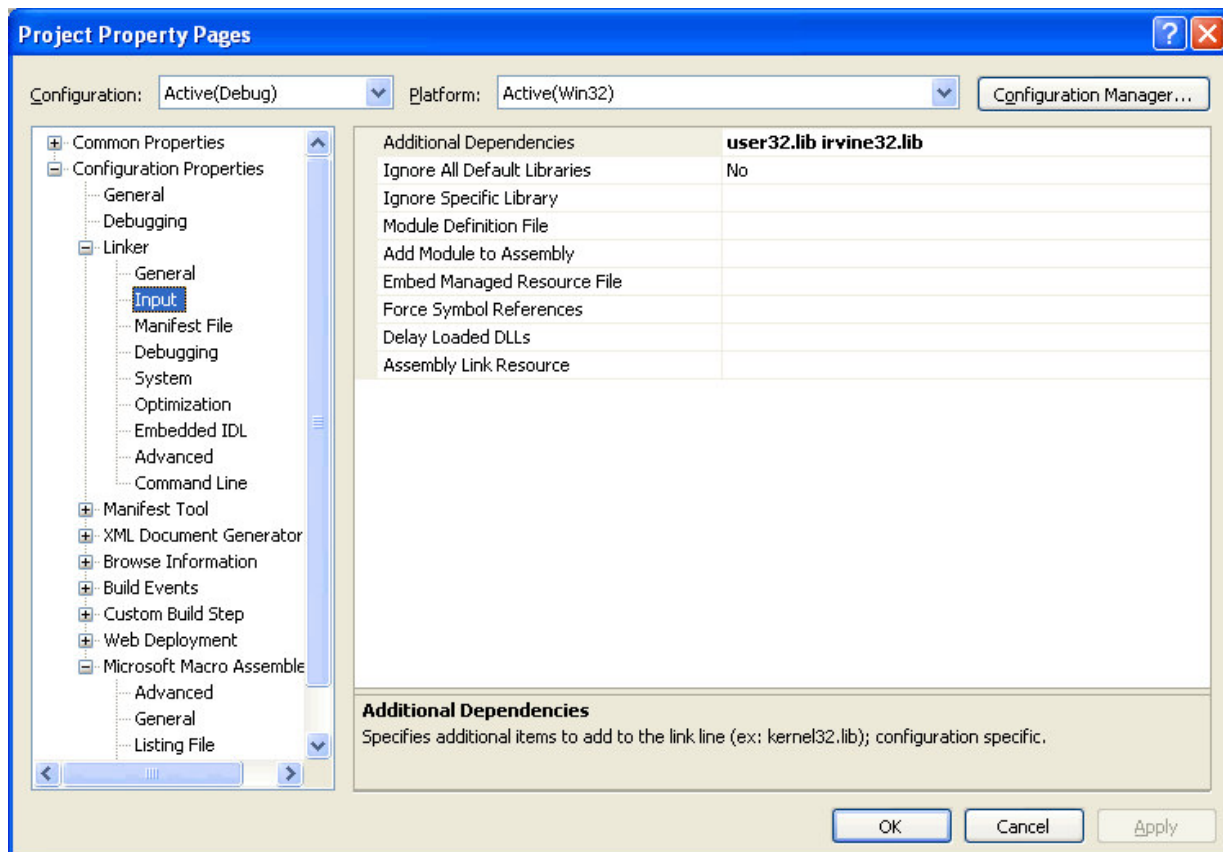
Assuming that our sample project is still open, select **Project Properties** from the Project menu. Expand the entry under **Configuration Properties**. Then expand the entry named **Microsoft Macro Assembler 8.0**. This is what you should see:



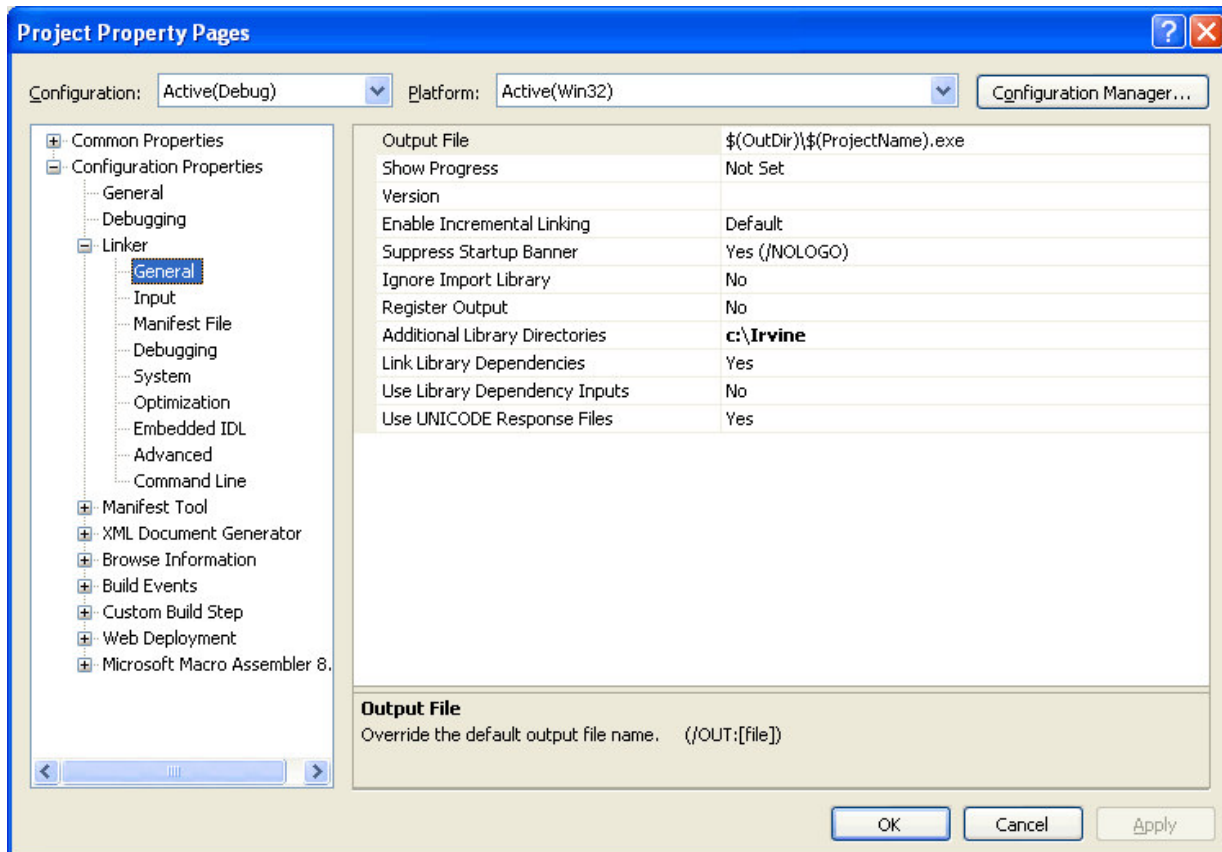
Click the entry named **General** under **Microsoft Macro Assembler 8.0**. Notice that the **Include Paths** option has been set to the c:\Irvine directory. This tells the assembler where to find files having a filename extension of ".inc". Here is a sample:



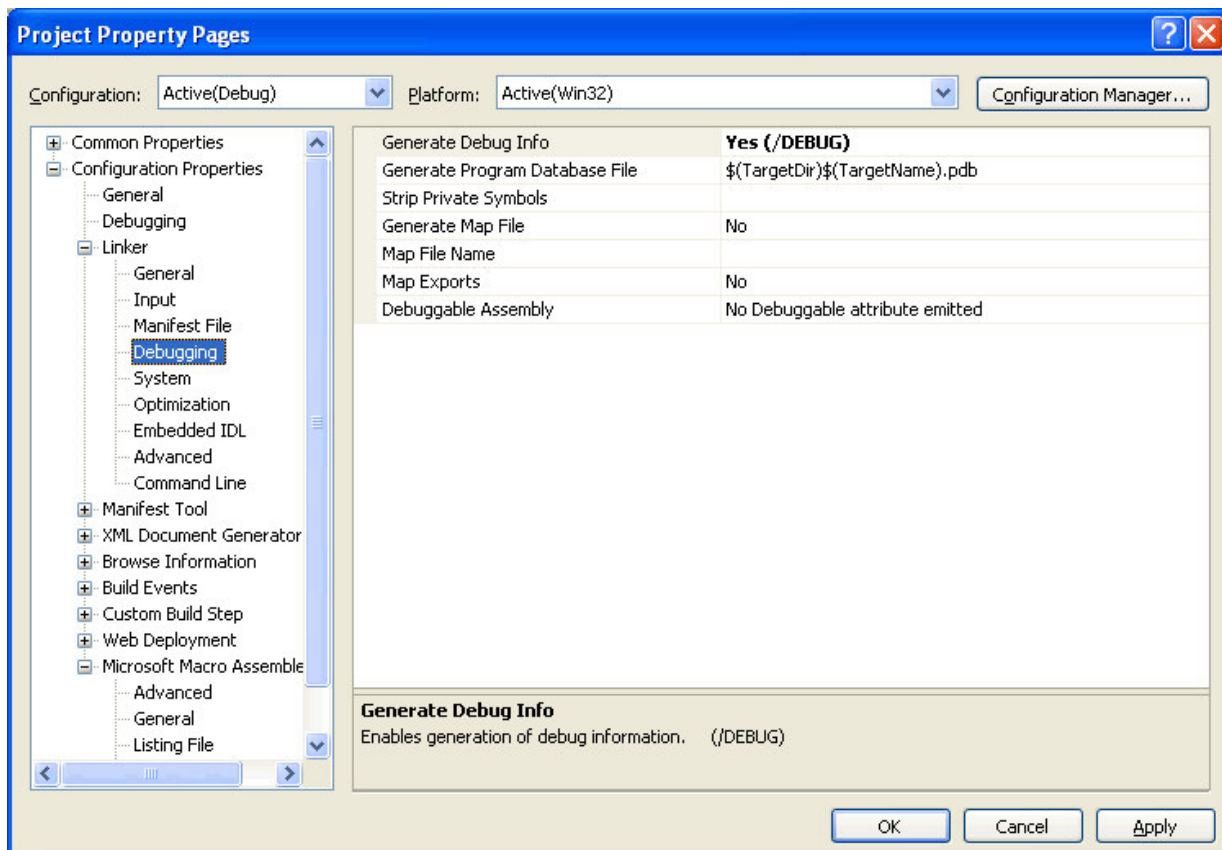
Find the **Linker** entry under **Configuration Properties**. Select the **Input** entry, and notice that two filenames have been added to the **Additional Dependencies** entry. The **user32.lib** file is a standard MS-Windows file. The **irvine32.lib** file is the link library file supplied with this book. There must be at least one space separating the file names:



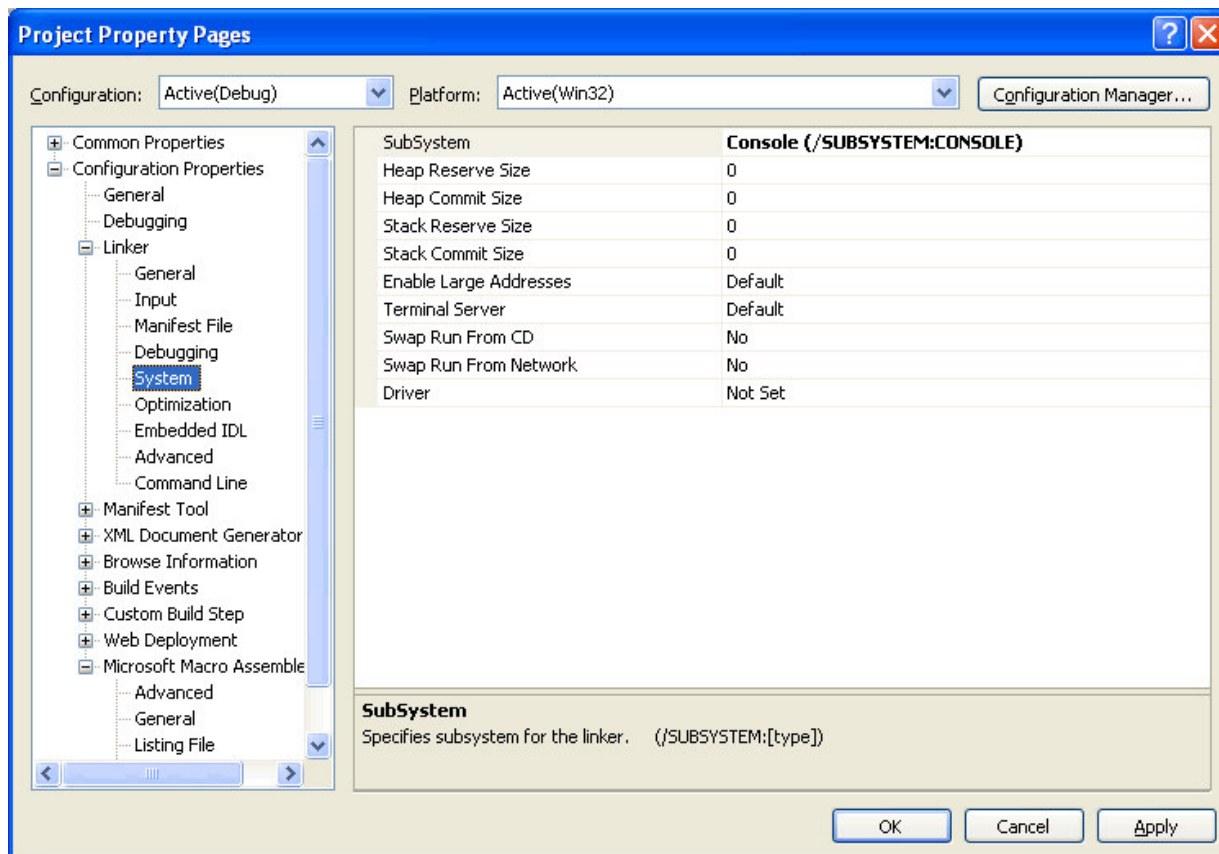
Next, select **Linker** under **Configuration Properties**, and then select **General**. The **Additional Library Directories** option equals **c:\Irvine**, so the linker can find the Irvine32.lib library file:



Select **Linker** under the **Configuration Properties** and select **Debugging**. Notice that the **Generate Debug Info** option is set to **Yes**:



Select **System** under the **Linker** entry. Notice that the SubSystem option has been set to **Console**:



We use the Console setting because it's easy for assembly language programs write output to a text console (Command) window. This is the window you see when running `cmd.exe` from the Start > Run menu in Windows.

Click the OK button to close the Project Property Pages window.

[Return to top](#)

Generating a Source Listing File

Prior to 7/26/06, the sample Visual Express projects in the book's download file do not generate source listing files. Here's how to change that behavior in a single project:

Open the project. From the menu, select **Project**, select **Project Properties**. In the list box, select **Microsoft Macro Assembler**, then select **Listing File**. Set the **Assembled Code Listing file** option to **\$(InputName).lst**.

[Return to top](#)

Creating a Project from Scratch

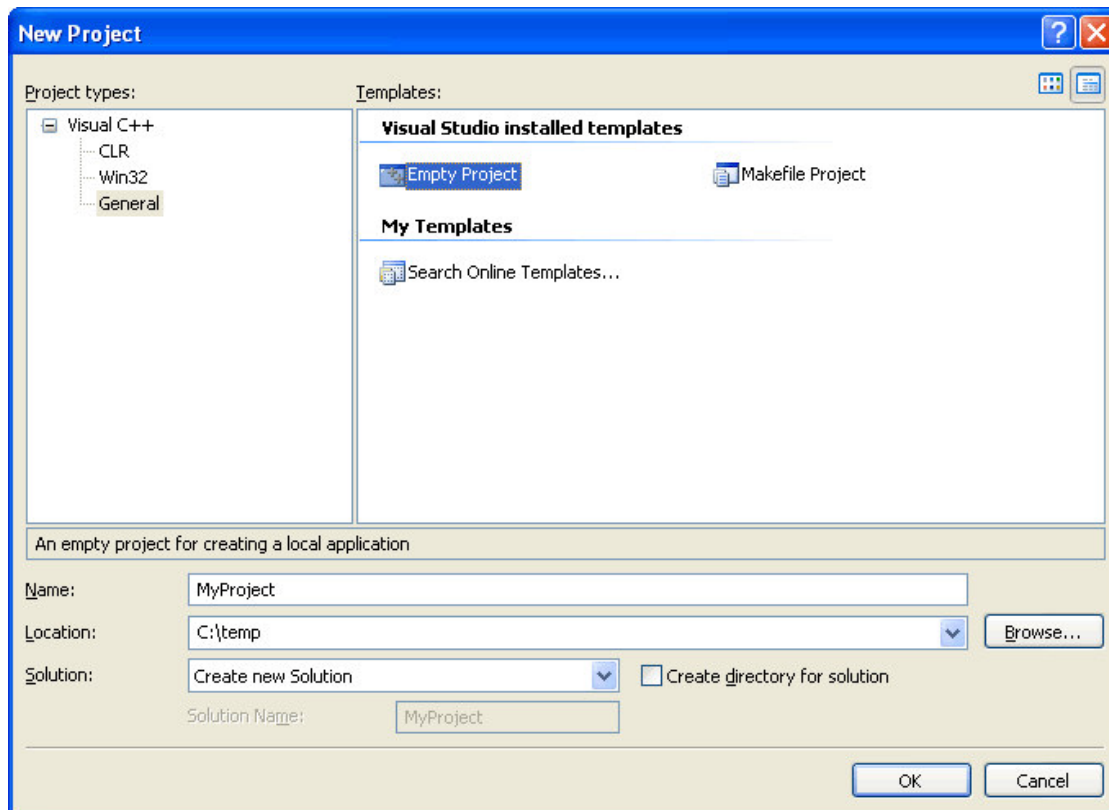
You do not have to create your own projects completely by yourself. Quite frankly, it's a lot of work.

We've placed a copy of the **Project_sample** project in each folder of the book's example programs. You can just add your own program to one of these projects. But here's how to create your own:

You can name a project anything you want, of course, but we will assume your project is named **MyProject** in the following examples, and that you will save it in the **c:\temp** directory. The commands are a little different, depending on which software you use:

Visual C++ Express

1. Select **New** from the File menu, and select **Project**.
2. In the New Project window, select **General**, and select **Empty Project** as the project type:

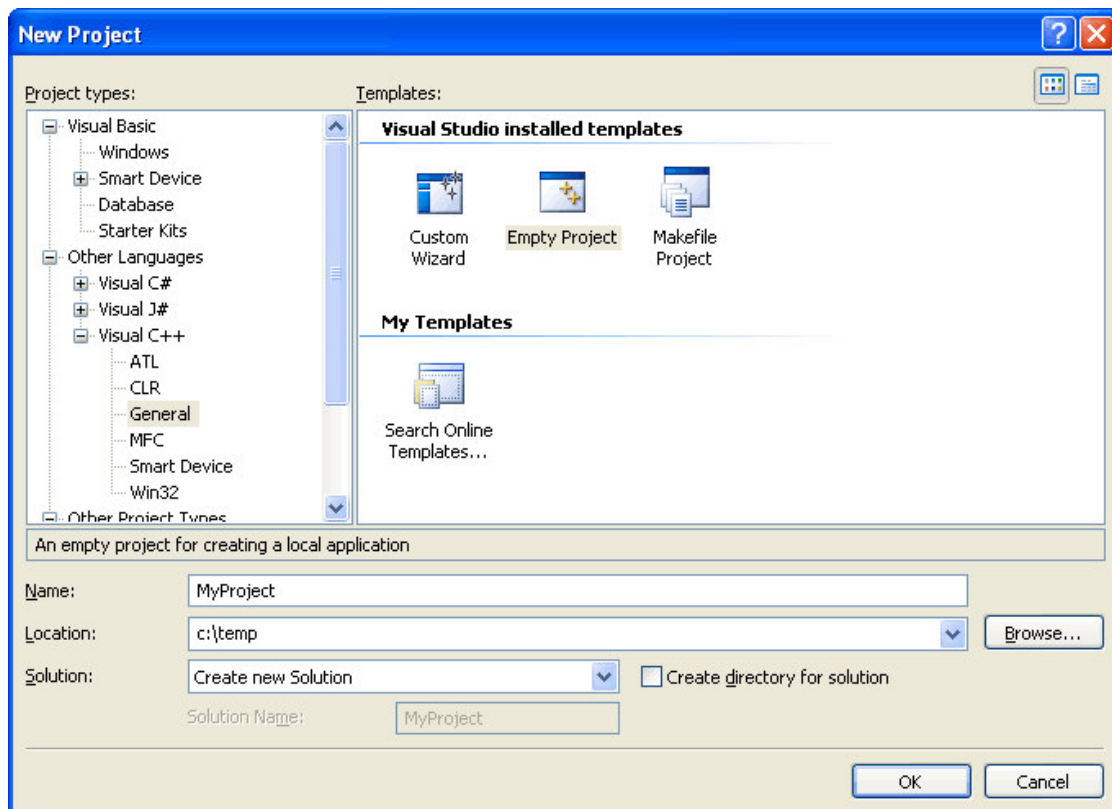


You probably will want to leave the *Create directory for solution* option unchecked.

3. Click the OK button to create the empty project.

Visual Studio

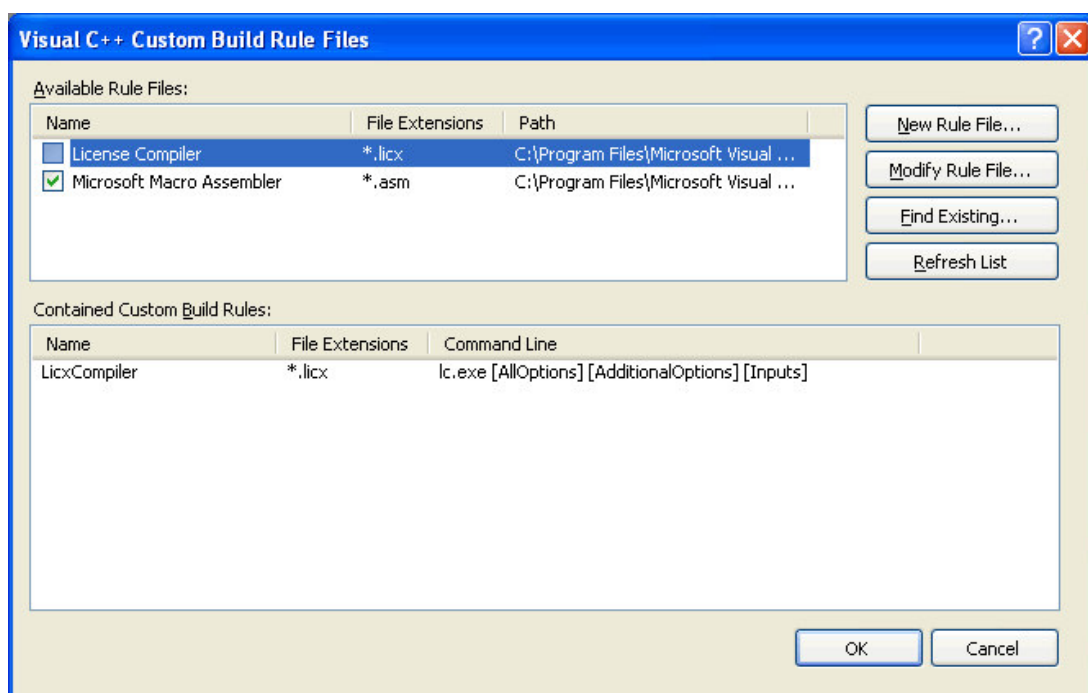
1. Select **New Project** from the **File** menu.
2. In the New Project dialog (shown in the image below), select **Other Languages**, select **Visual C++**, select **General**, and select **Empty Project**.
3. Give a name to the project and select the location. Do not change any other options.
4. Click the OK button.



You probably will want to leave the *Create directory for solution* option unchecked.

Both Visual Studio and Visual C++ Express

1. Use Windows Explorer or My Computer to copy the file **main.asm** from the \examples \Project_sample folder into the project folder you must created. (In our example, the folder is named MyProject).
2. Back in Visual Studio or Visual C++ Express, right click your project name in the Solution Explorer window. Select Add, select Existing Item, and select main.asm. (The file may not show up until you input *.asm in the filename box.) Click the Add button to add it to your project.
3. Select **Custom Build Rules** from the Project menu. You will see a list of Rule files, which will vary depending on which software you are using. Place a check next to Microsoft Macro Assembler:



Click the OK button to close this window.

7. Next, you need to add some customizations. We will assume you installed the book's files in the c:\Irvine directory. Make all changes shown in the [Project Properties Settings](#) section of this document. If you installed the book's sample programs in some other location than c:\Irvine, you'll need to make appropriate changes to the project properties.

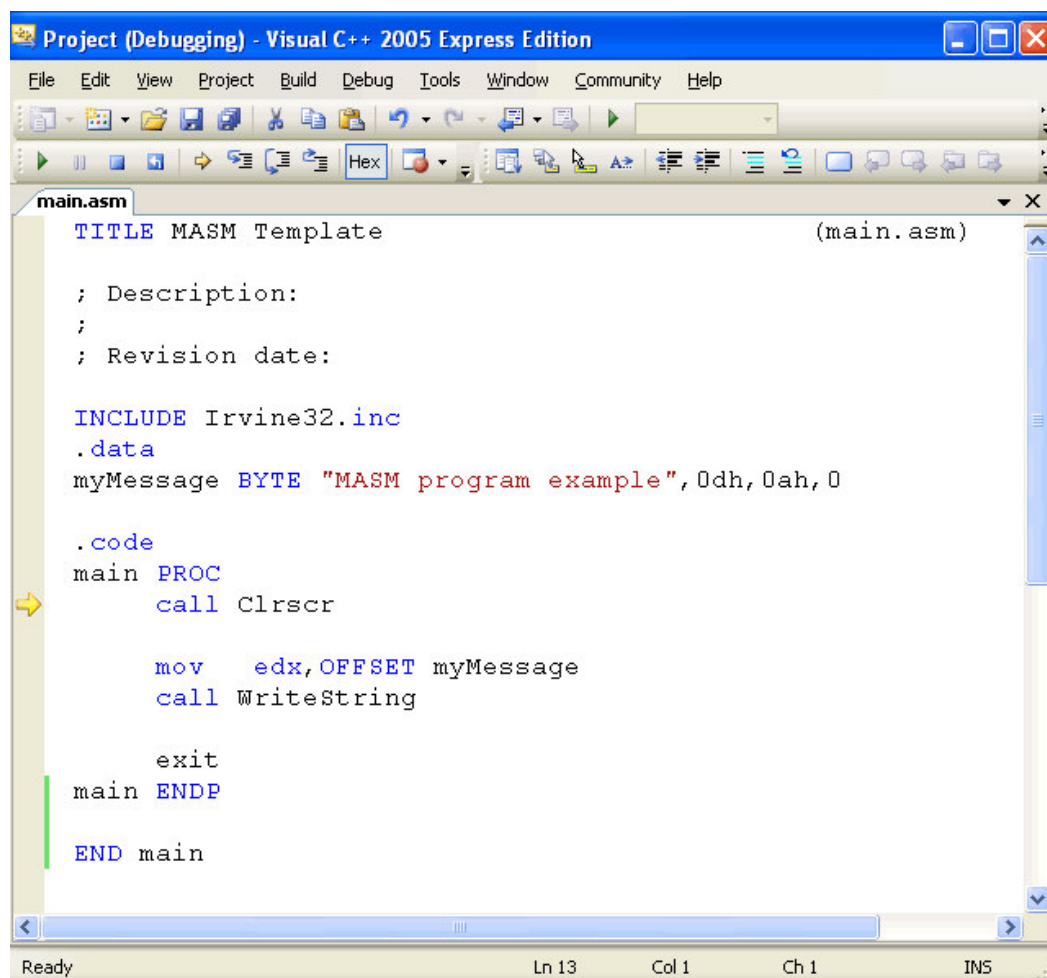
9. Select **Build Solution**. If your Output window is similar to the following message, you did everything right:

```
1>----- Build started: Project: MyProject, Configuration: Debug Win32
1>Linking...
1>Embedding manifest...
1>Build log was saved at "file:///c:/temp/MyProject/Debug/BuildLog.htm"
1>MyProject - 0 error(s), 0 warning(s)
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped
```

[Return to top](#)

MASM syntax highlighting

When a text editor uses syntax highlighting, language keywords, strings, and other elements appear in different colors. Visual Studio and Visual C++ Express can highlight MASM reserved words and strings, as shown in the following example:

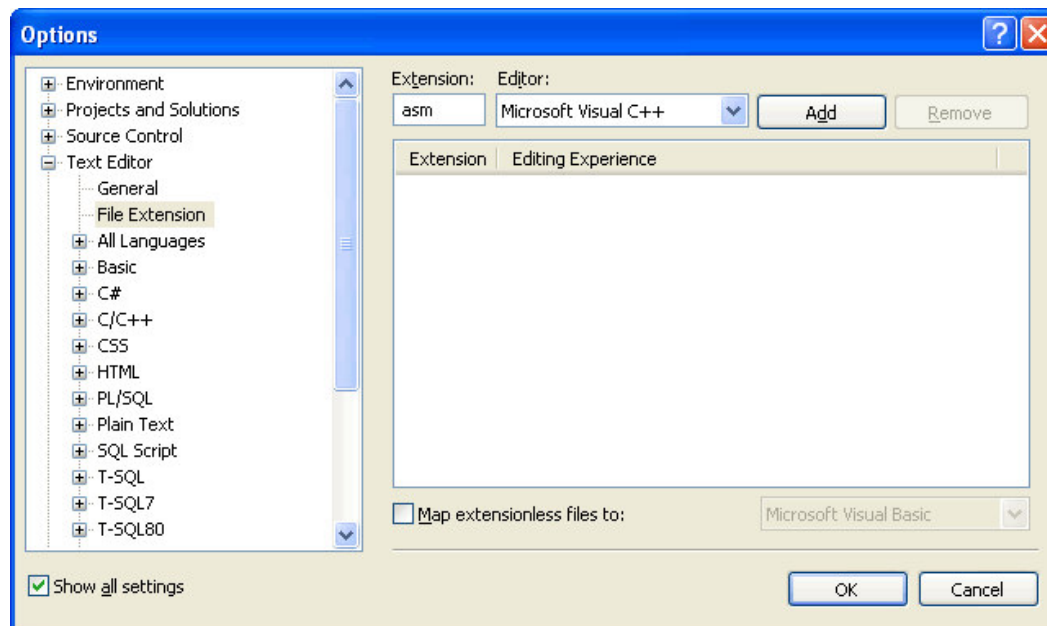


This won't happen automatically, but you can create a syntax definition file named UserType.dat that contains MASM keywords. Then when Visual Studio (or Visual C++ Express) starts, it reads the syntax

file and highlights MASM keywords.

Here are the required steps to set up MASM syntax highlighting in Visual Studio or Visual C++ Express:

- 1) Download the [Ustertype.dat](#) file given here to a folder in which you have read/write permissions. If you are using Windows Vista, download to My Documents, or C:\temp, or any folder that doesn't have security restrictions.
- 2) Copy Ustertype.dat to the C:\Program Files\Microsoft Visual Studio 9.0\Common7\IDE folder. If you are using Windows Vista, it will display a verification window before copying the file.
- 3) Open Visual Studio or Visual C++ Express, select **Options** from the Tools menu, select **Text Editor**, and select **File Extension**. On the right side of the dialog (shown below), enter **asm** as the extension, select **Microsoft Visual C++** from the Editor list, and click the **Add** button. Click the **OK** button to save your changes.



Close Visual Studio and restart it. Open your project and display an ASM file. You should see syntax highlighting in the editor.

[Return to top](#)

Assembling, Linking, and Debugging with a Batch File

Many people like to use a *Windows batch file* to assemble and link programs. A batch file is a text file containing a sequence of commands that execute as if they had been typed at the command prompt. In fact, they are powerful enough to contain variables, loops, IF statements, and so on.

The easiest way to run a batch file is to first open a Command window and then type the name of the batch file (along with arguments) at the command prompt. To open a Command window, you must execute a program named **cmd.exe**. We will make that easy for you.

[Right-click here to download a zip file](#) containing the following items:

- **A shortcut to cmd.exe**, which opens a Command window in the current directory
- **asm32.bat**, a batch file for assembling and linking programs

- **main.asm**, a sample assembly language program

Extract this ZIP file into the c:\Irvine\Examples directory on your computer. Then do the following:

- Copy asm32.bat to any directory on your system path. If you're using Windows XP, you can save it in the **c:\Windows** folder. In Windows Vista, you will need administrator privileges to do that. Having asm32.bat on your system path permits **asm32** to be a valid command whenever you type its name at the command prompt.
- Double-click the **cmd.exe** shortcut. A Command window should appear.
- At the command prompt in this window, execute the asm32 batch file and display its help information by typing the following command and pressing **Enter**:

```
asm32
```

This file assembles, links, and debugs a single assembly language source file. Before using it, install Visual Studio 2008 in the following directory:

```
C:\Program Files\Microsoft Visual Studio 9.0
```

Next, install the Irvine 5th edition link libraires and include files in the following directory: C:\Irvine

Finally, copy this batch file to a location on your system path. We recommend the following directory:

```
C:\Program Files\Microsoft Visual Studio 9.0\VC\bin
```

Command-line syntax:

```
asm32 [/H | /h | -H | -h] -- display this help information
```

```
asm32 filelist -- assemble and link all files
```

```
asm32 /D filelist -- assemble, link, and debug
```

```
asm32 /C filelist -- assemble only
```

<filelist> is a list of up to 5 filenames (without extensions), separated by spaces. The filenames are assumed to refer to files having .asm extensions. Command-line switches are case-sensitive.

Type the following command to assemble and link a source file named **main.asm**:

```
asm32 main
```

You should see the following messages:

```
Assembling: main.asm
```

```
The file main.obj was produced.
```

```
.....
```

```
Linking main.obj to the Irvine32, Kernel32, and User32 libraries.
```

```
The file main.exe was produced.
```

.....

In fact, several files were produced.

- main.obj - the object file
- main.ilc - incremental link status file
- main.pdb - debug symbol file

If there were errors in the program, you would see error messages generated by the assembler. Here is an example:

```
Assembling: main.asm
main.asm(9) : error A2008: syntax error : myMessage
main.asm(15) : error A2006: undefined symbol : myMessage
```

You would then open the main.asm file with a text editor (such as Notepad), fix the errors, and run the asm32 batch file again.

Although we used a file named main.asm in this example, the asm32.bat batch file will work for any assembly language file, regardless of the name. The only requirement is that your assembly language source file have a **.asm** filename extension.

Assembling Programs in Other Directories

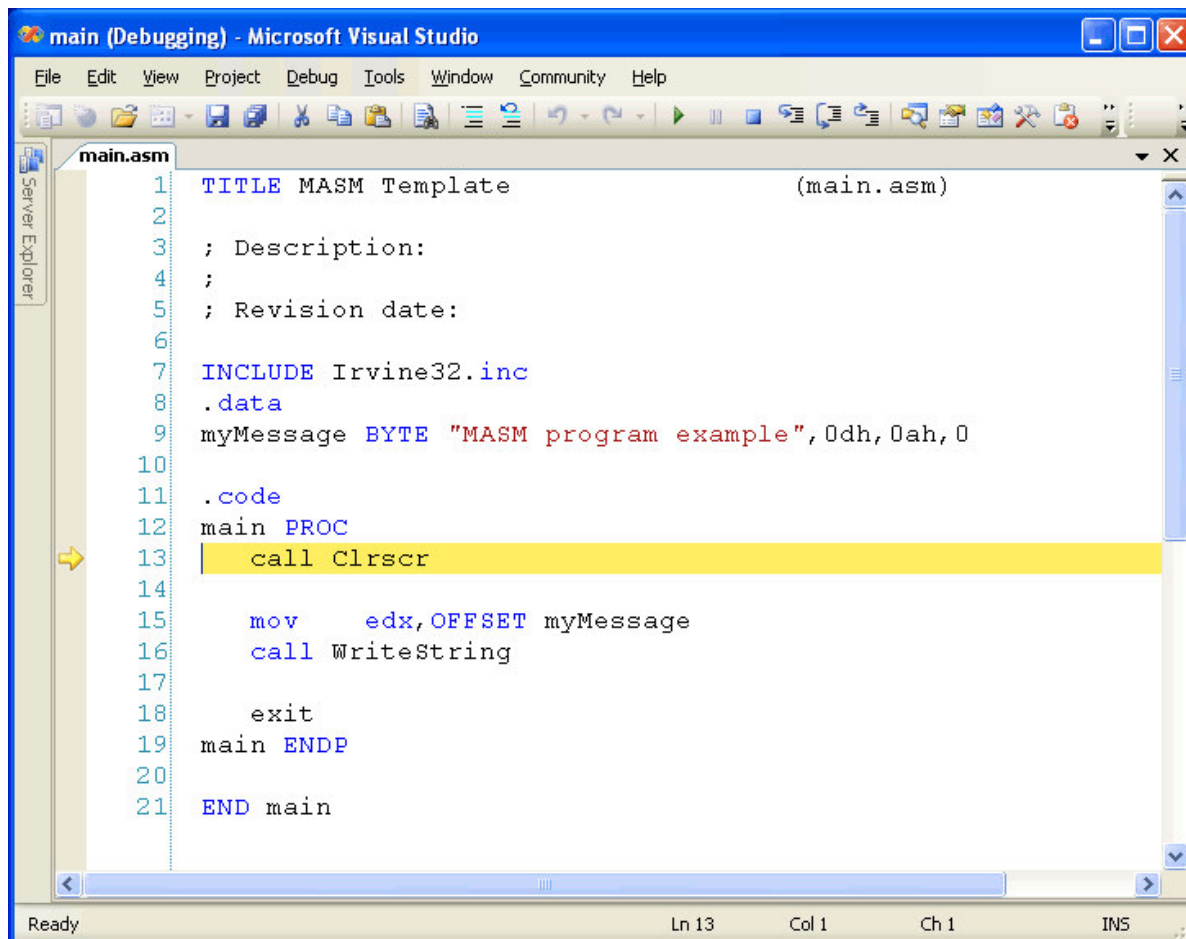
No doubt, you will want to assemble programs in various different disk folders, not just the batch_sample folder used in the foregoing example. All you need to do is copy the **cmd.exe** shortcut we gave you to your working directory, where your assembly language source files are located. When you double-click to run the shortcut, it will open a Command window in the current folder.

Assembling, Linking, and Debugging

In addition to assembling and linking, you can use the asm32.bat file to launch your program in the Visual Studio debugger. Try the following command:

```
asm32 /D main
```

If the program assembles and links with no errors, your program should load in Visual Studio. The first time you do this with a new program, the source code will not appear. All you have to do is press the **F10 key** to begin debugging, and your program should appear with a yellow band across the first executable line:



(Depending on how Visual Studio is configured, you might have to press F8 to do the same thing.)

From here, you can step through the program. When you get to the call to WriteString, you can even trace into its code by pressing the F11 key (trace to). When you finish, close Visual Studio.

From this time on, when you load the same program in the Visual Studio debugger, your source code will appear right away.

Assembling without Linking

Occasionally, you may want to assemble programs but not link them. This happens, for example, when you are creating a multimodule project and you want to assemble each asm file into an obj file separately before linking them into the final exe program. Or, you might be assembling a module to be inserted into a link library (like Irvine32.lib).

To assemble a source file only, insert the /C option before the name of the file being assembled:

```
asm32 /C main
```

You should see the following output:

```
Assembling: main.asm
```

```
The file main.obj was produced.
```

```
.....
```

If you are interested in learning more about how batch file commands work, here are some reference links we found:

- Allenware.com: [Batch file tutorial](#)
- Microsoft TechNet article: [Creating Truly Powerful Batch Files](#), by Brien Posey
- Microsoft TechNet article: [Using Batch Files in Windows NT](#), by Troy Thompson

Links go out of date quickly, but you can google for *Windows batch files* and get plenty of hits.

[Return to top](#)