APPENDIX A

Using Visual C ++

In this appendix, you learn how to create simple programs using Visual C++ and Microsoft Macro Assembler (MASM). With this knowledge, you'll be able to use and experiment with the sample code that's discussed in this book. You'll also be able to create your own programs using Visual C++ and MASM.

Visual Studio uses entities called solutions and projects to help simplify application development. A solution is a collection of one or more projects that are used to build an application. Projects are container objects that help organize an application's files. A Visual Studio project is usually created for each buildable component of an application (e.g. executable file, dynamic-linked library, static library, and so on). You can open and load any of this book's sample programs into the Visual Studio development environment by double-clicking on its solution (`.sln`) file.

A standard Visual C++ project includes two solution configurations named Debug and Release. As implied by their names, these configurations are used to support separate executable builds for initial development and final release. A standard Visual C++ project also incorporates solution platforms. The solution platform named Win32 contains the settings that are needed to build a 32-bit executable file. A solution platform named x64 can optionally be added to build a 64-bit executable file.

All of the examples in this appendix were created using Visual Studio Professional 2013 with Update 3 using the following settings:

- Visual C++ Development Settings
- Visual C++ 6 Keyboard-Mapping Scheme

Visual Studio uses keyboard-mapping schemes to define key sequences and shortcuts for commonly-performed operations. You can create a Visual C++ application using one of the alternate development settings or keyboard mapping schemes (such as C#). However, the menus and keyboard sequences will differ from what's used in this appendix. You can also use Visual Studio Express 2013 with Update 3 for Windows Desktop, which can be freely downloaded from the following Microsoft website: `http://msdn.microsoft.com/en-us/vstudio`. Visual Studio Express does not support language-specific development settings, but you can change its default keyboard mapping scheme to Visual C++ 6 as follows:

1. Select Tools | Options.
2. In the Options dialog box tree control, select Environment | Keyboard.

3. In the keyboard-mapping scheme drop-down list, select **Visual C++ 6**.
4. Click on OK.

Example 1: A Simple Win32 Project

In this example, you create a simple Visual C++ Win32 project that uses an x86-32 assembly language function.

Create the Project

Use the following steps to create a Visual C++ project:

1. Start Visual Studio 2013.
2. Select File | New Project.
3. In the New Project dialog box tree control, select Installed | Templates | Visual C++ | Win32.
4. Select **Win32 Console Application** for the project type.
5. In the Name text box, enter `Example1`.
6. In the Location text box, enter a folder name for the project location. You can also use the Browse button to choose a folder.
7. Verify that the New Project dialog box settings match the settings that are shown in Figure A-1 (the location name can be different.) The default location for Visual Studio solutions is the subfolder named `Visual Studio 2013\Projects` that's located in your default documents folder.

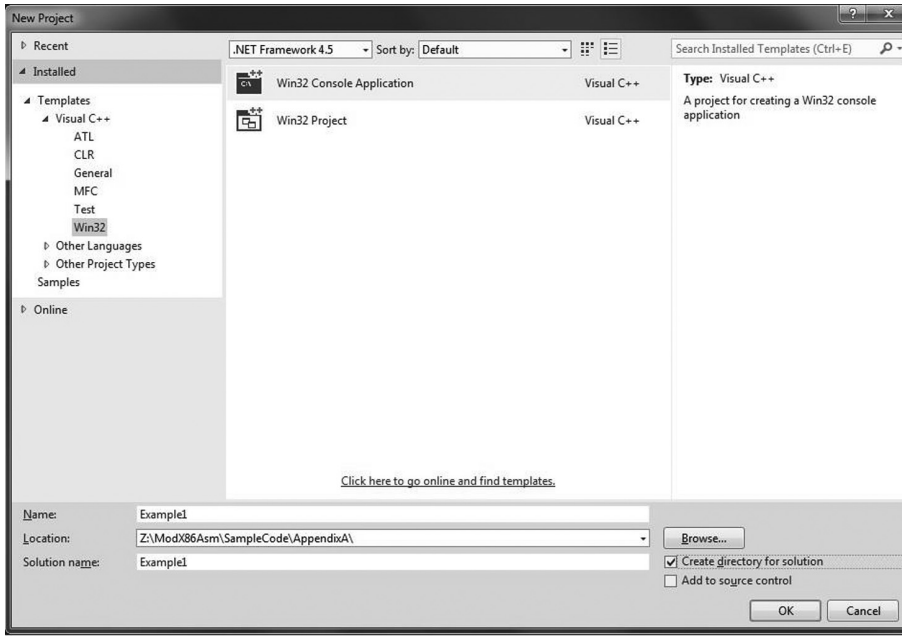

Figure A-1. *New Project dialog box settings*

8. Click on OK.
9. In the Welcome to the Win32 Application Wizard dialog box, click on Next.
10. Verify that the settings on the Application Settings dialog box match the settings shown in Figure A-2.

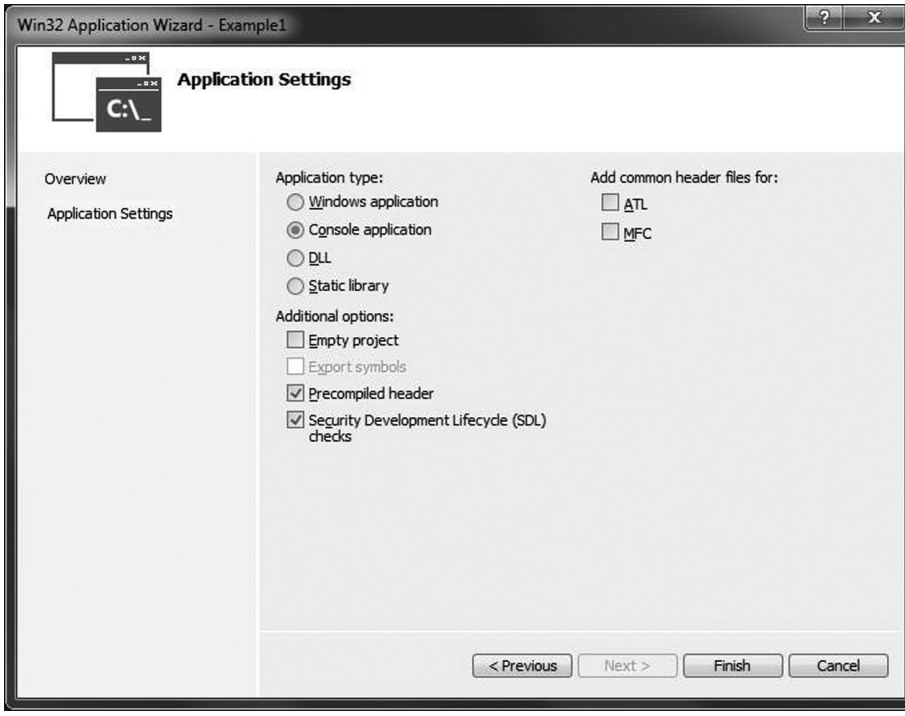

Figure A-2. *Application Settings dialog box*

11. Click on Finish.

Enable Support for MASM

Use the following steps to enable support for Microsoft Macro Assembler:

1. Select View | Solution Explorer.
2. In the Solution Explorer tree control, right-click on `Example1` and select Build Dependencies | Build Customizations.
3. In the Visual C++ Build Customization Files dialog box, check **masm(.targets, .props)**, as shown in Figure A-3.

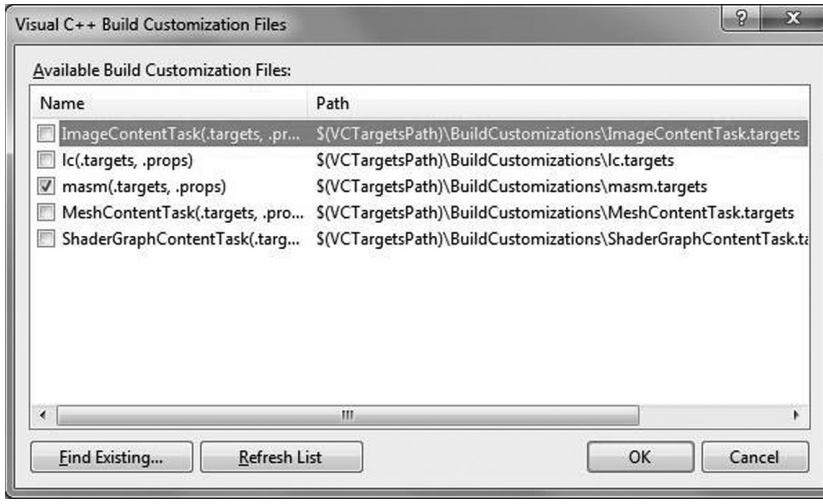

Figure A-3. *Visual C++ Build Customization Files dialog box*

4. Click on OK.

Add an Assembly Language File

Use the following steps to add an assembly language source code file (`.asm`) to a Visual C++ project:

1. In the Solution Explorer tree control, right-click on `Example1` and select Add | New Item.
2. In the Add New Item dialog box tree control, select Installed | Visual C++ | Code.
3. Select **C++ File (.cpp)** as the file type.
4. In the Name text box, change the filename to `Example1_.asm`, as shown in Figure A-4.

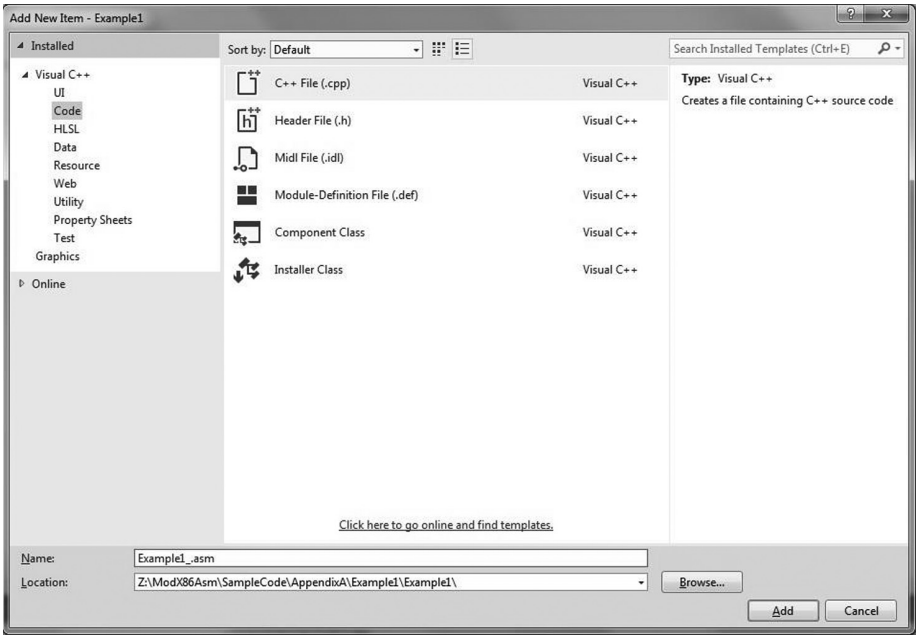

Figure A-4. *Add New Item dialog box*

5. Click on Add.

Set the Project Properties

Use the following steps to set a project's properties:

1. In the Solution Explorer tree control, right-click on `Example1` and select **Properties**.
2. In the Property Pages dialog box, select **All Configurations** for Configuration.
3. In the tree control, select Configuration Properties | Microsoft Macro Assembler | Advanced.
4. Click on **Use Safe Exception Handlers**. Then select **Yes (/safeseh)** for Use Safe Exception Handlers, as shown in Figure A-5. (This switch enables MASM to generate object modules that are compatible with the C++ compiler and linker.)

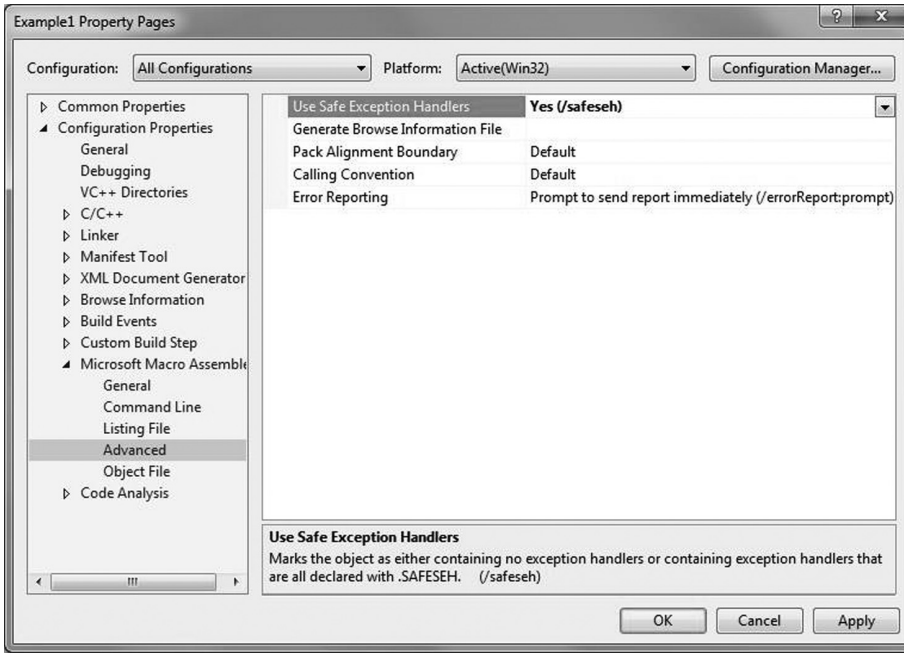

Figure A-5. *Microsoft Macro Assembler advanced properties*

5. Optional. Use the following steps to generate a MASM listing file:

    a. In the tree control, select Configuration Properties | Microsoft Macro Assembler | Listing File.

    b. Select **Yes (/sg)** for Enable Assembly Generated Code Listing.

    c. Enter the text `$(IntDir)\%(Filename).lst` in Assembled Code Listing File, as shown in Figure A-6 (the macro text specifies the project's intermediate directory, which is a subfolder of the folder that contains the `.cpp` and `.asm` files).

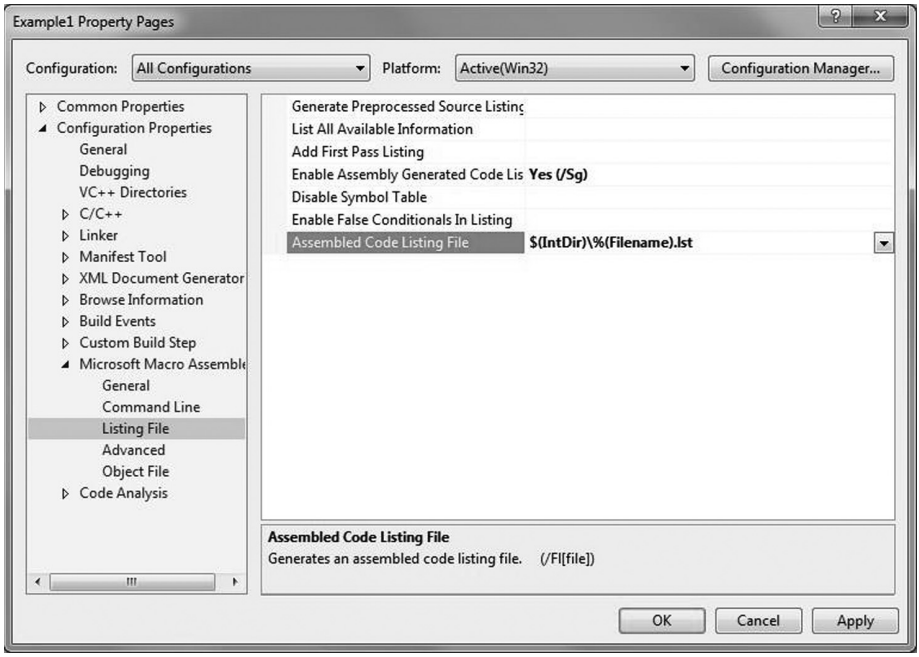

Figure A-6. *Microsoft Macro Assembler Listing File properties*

6. Click on OK.
7. Select File | Save All.

Edit the Source Code

Use the following steps to edit the project source code:

1. In the Editor window, click on the tab named `Example1.cpp`.
2. Edit the C++ source code so that it matches the content of Listing A-1.
3. Click on the tab named `Example1_.asm`.
4. Enter the x86-32 assembly language source code that's shown in Listing A-2.
5. Select File | Save All.

Listing A-1. Example1.cpp

```
#include "stdafx.h"

extern "C" int CalcResult1_(int a, int b, int c);

int _tmain(int argc, _TCHAR* argv[])
{
    int a = 30;
    int b = 20;
    int c = 10;
    int d = CalcResult1_(a, b, c);

    printf("a: %4d b: %4d c: %4d\n", a, b, c);
    printf("d: %4d\n", d);
    return 0;
}
```

Listing A-2. Example1_.asm

```
        .model flat,c
        .code

; extern "C" int CalcResult1_(int a, int b, int c);

CalcResult1_ proc
        push ebp
        mov ebp,esp

        mov eax,[ebp+8]                 ;eax = a
        mov ecx,[ebp+12]                ;ecx = b
        mov edx,[ebp+16]                ;edx = c

        add eax,ecx                     ;eax = a + b
        imul eax,edx                    ;eax = (a + b) * c

        pop ebp
        ret
CalcResult1_ endp
        end
```

Build and Run the Project

Use the following steps to build and run the project:

1. Select Build | Build Solution.
2. If necessary, fix any reported compiler or assembler errors and repeat Step 1.

3. Select Debug | Start Without Debugging.
4. Verify that the output matches the console window shown in Figure A-7.

Figure A-7. *Console window output for Example1*

5. Press Enter to close the console window.
6. Select File | Exit to close Visual Studio.

If you build and run a program after making source code changes, Visual Studio may display a project out-of-date warning dialog box similar to the one that's shown in Figure A-8. You can disable this warning by checking the **Do not show this dialog again** box. Then click on Yes to complete the build and run the program.

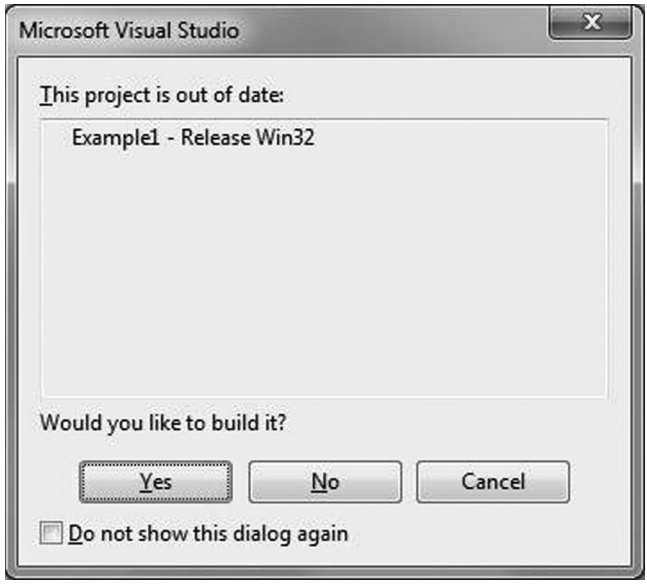

Figure A-8. *Visual Studio out-of-date warning dialog box*

Example 2: Visual C++ Debugger

In this example, you learn how to perform basic operations using the Visual C++ debugger.

Create and Build the Project

Use the following steps to create and build the project:

1. Start Visual Studio.
2. Create a new project named `Example2`. Follow the steps in the Example1 sections entitled "Create the Project" through "Set the Project Properties." The source code files should be named `Example2.cpp` and `Example2_.asm`.
3. Using the Visual Studio editor, enter the source code for `Example2.cpp` and `Example2_.asm` shown in Listings A-3 and A-4.
4. Select Build | Build Solution.
5. If necessary, fix any reported compiler or assembler errors and repeat Step 4.
6. Select Debug | Start Without Debugging.
7. Verify that the output of your program matches the console window shown in Figure A-9.

Listing A-3. Example2.cpp

```
#include "stdafx.h"

extern "C" void CalcResult2_(int a, int b, int c, int* quo, int* rem);

int _tmain(int argc, _TCHAR* argv[])
{
    int a = 75;
    int b = 125;
    int c = 7;
    int quo, rem;

    CalcResult2_(a, b, c, &quo, &rem);

    printf("a: %4d b: %4d c: %4d\n", a, b, c);
    printf("quo: %4d rem: %4d\n", quo, rem);
    return 0;
}
```

Listing A-4. Example2_.asm

```
        .model flat,c
        .code

; extern "C" void CalcResult2_(int a, int b, int c, int* quo, int* rem);

CalcResult2_ proc
        push ebp
        mov ebp,esp

; Calculate (a + b) / c
        mov eax,[ebp+8]                 ;eax = a
        mov ecx,[ebp+12]                ;ecx = b
        add eax,ecx                     ;eax = a + b

        cdq                             ;edx:eax contains dividend
        idiv dword ptr [ebp+16]         ;eax = quotient, edx = rem

        mov ecx,[ebp+20]                ;ecx = ptr to quo
        mov dword ptr [ecx],eax         ;save quotent
        mov ecx,[ebp+24]                ;ecx = ptr to rem
        mov dword ptr [ecx],edx         ;save remainder

        pop ebp
        ret
CalcResult2_ endp
        end
```

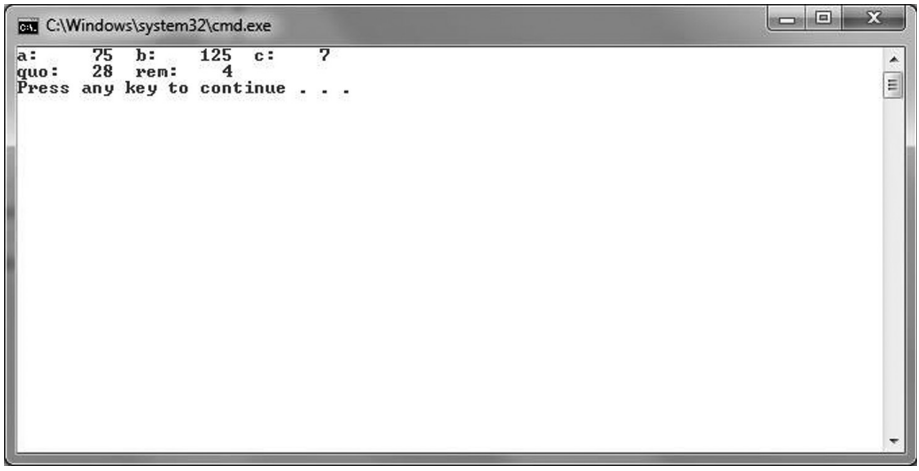

Figure A-9. *Console window output for Example2*

Using the Visual C++ Debugger

Use the following steps to debug the project:

1. In the Editor window, click on the tab named `Example2.cpp`.
2. Move the cursor to (or click on) the C++ statement `CalcResult2_(a, b, c, &quo, &rem);`.
3. Press Ctrl+F10. This key builds the project (if necessary), starts the Visual C++ debugger, and executes code until the cursor line is reached.
4. Press F11. This key steps into the function `CalcResult2_`.
5. Press Alt+5 to open the Registers window.
6. If necessary, right-click in the Registers window and select **CPU**. This displays the contents of the general-purpose registers.
7. If necessary, right-click in the Registers window and select **Flags**. This displays the current state of the status flags, as shown in Figure A-10. Note that the Visual C++ debugger uses the following names for the x86 status flags:

   OV = EFLAGS.OF

   UP = EFLAGS.DF

   EI = EFLAGS.IF

   PL = EFLAGS.SF

ZR = EFLAGS.ZF

AC = EFLAGS.AF

PE = EFLAGS.PF

CY = EFLAGS.CY

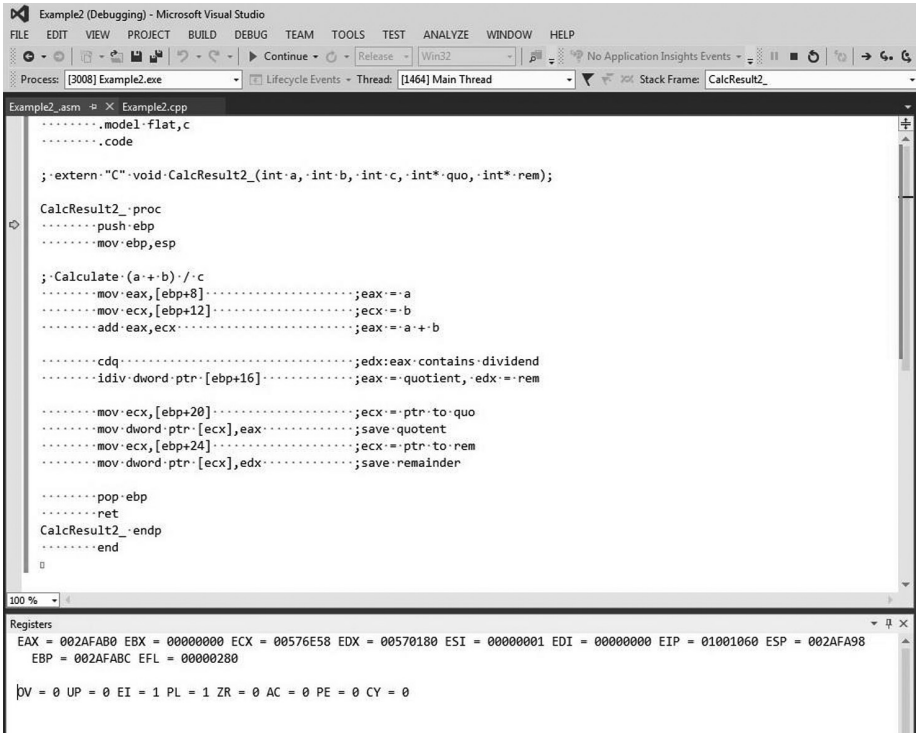

Figure A-10. *Visual C++ Registers window*

8. Press F10. This key executes the next instruction. Note that the debugger highlights in red all registers that were modified by the execution of the `push ebp` instruction.

9. Press F10 until the yellow arrow points to the `idiv dword ptr [ebp+16]` instruction. Note that register pair EDX:EAX contains the value 0x00000000:0x000000C8 (200). Also note the line below the status flags, which shows the contents of the memory location `[ebp+16]` (Figure A-11).

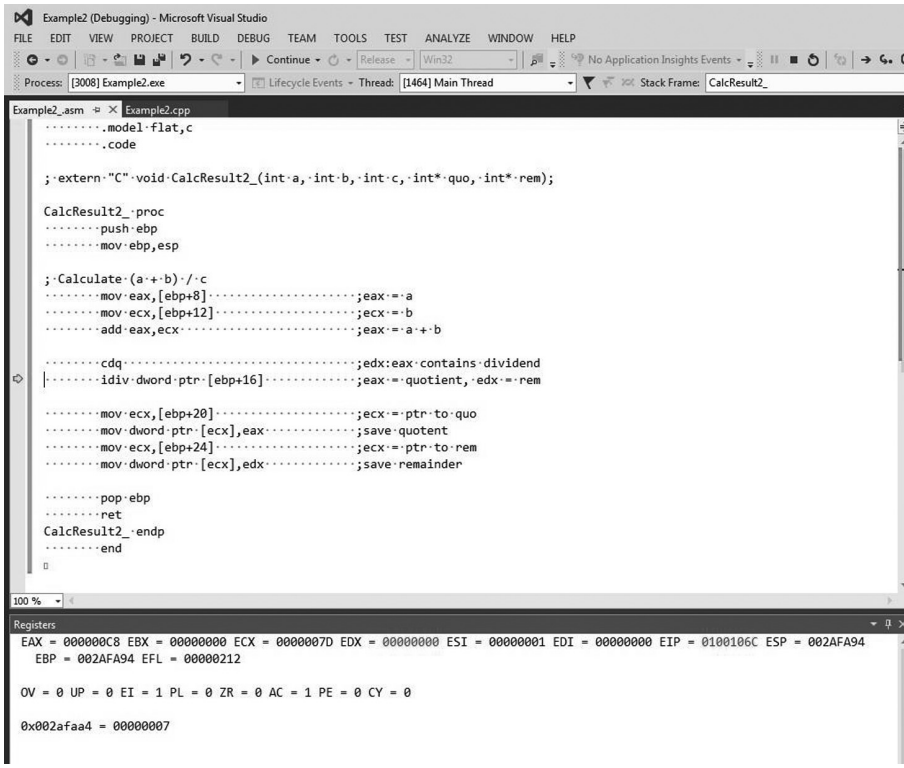

Figure A-11. *Program registers and flags prior to execution of the* `idiv dword ptr [ebp+16]` *instruction*

10. Press F10. This key executes the `idiv dword ptr [ebp+16]` instruction. Note that register EAX now contains the quotient (28 or 0x1C) and EDX contains the remainder (4).

11. Press Shift+F11. This key steps out of function `CalcResult2_` and returns to `_tmain`.

12. Press F10 until the yellow arrow points to the `return 0;` statement. You can open the Console window to view the output.

13. Press Shift+F5 to stop debugging.

14. Select File | Close Solution to close the solution or File | Exit to close Visual Studio.

Example 3: A Simple x64 Project

In this example, you create a simple Visual C++ x64 project that uses an x86-64 assembly language function.

Create the Project

Use the following steps to create the project:

1. Start Visual Studio.
2. Create a new project named `Example3`. Follow the steps in the Example1 sections entitled "Create the Project" through "Add an Assembly Language File." The source code files should be named `Example3.cpp` and `Example3_.asm`.
3. Select Build | Configuration Manager.
4. In the Configuration Manager dialog box (Figure A-12), select **<New...>** under Active Solution Platform.

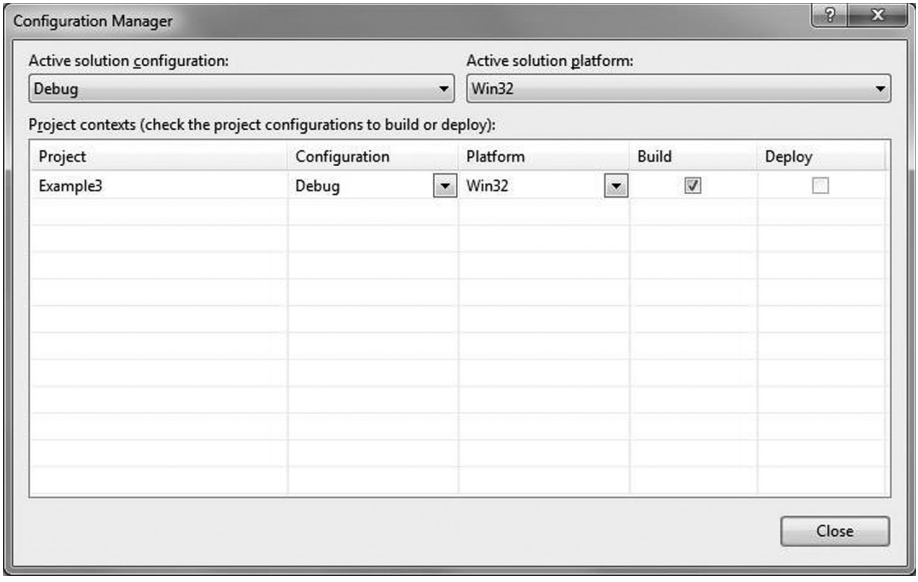

Figure A-12. *Configuration Manager dialog box*

5. In the New Solution Platform dialog box (Figure A-13), select **x64** under Type or Select the New Platform.

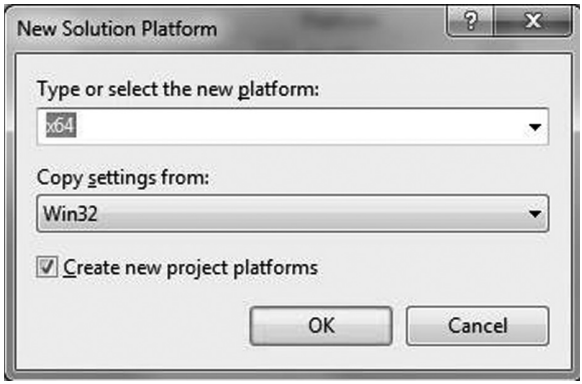

Figure A-13. *New Solution Platform dialog box*

6. Click on OK.
7. In the Configuration Manager dialog box, select **<Edit . . . >** under Active Solution Platform.
8. In the Edit Solution Platform dialog box, select **Win32** and click Remove. Then click Close.
9. In the Configuration Manager dialog box, click Close.

Edit Source Code, Build Project, and Run Program

Use the following steps to edit the source code, build the project, and then run the program:

1. Optional. Follow the steps in the Example1 section entitled "Setting Project Properties" to enable generation of a MASM listing file. ***Do not enable the Use Safe Exception Handlers option***.
2. Enter the text for `Example3.cpp` (Listing A-5) and `Example3_.asm` (Listing A-6).
3. Select Build | Build Solution (or press F7) to build the solution. If necessary, fix any reported compiler or assembler errors.
4. Select Debug | Start Without Debugging (or press Ctrl+F5) to run the program.

5. Verify that the output of your program matches the console window shown in Figure A-14.

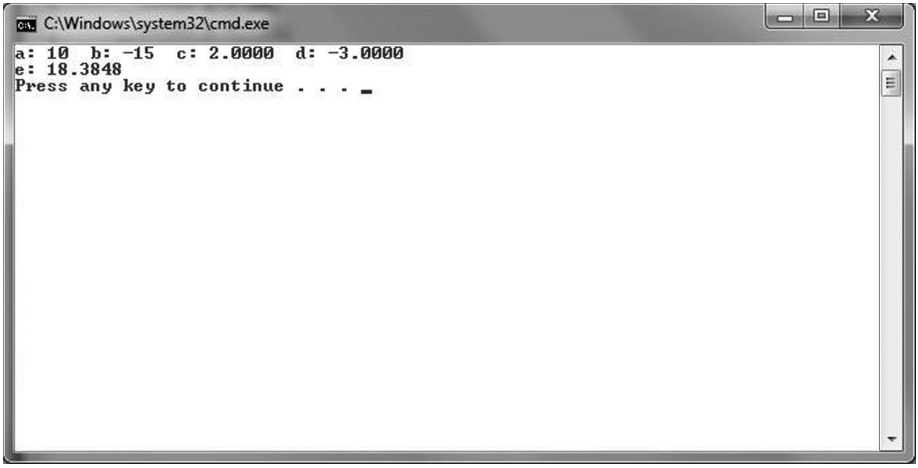

Figure A-14. *Console window output for Example3*

6. Select File | Close Solution to close the solution or File | Exit to close Visual Studio.

Listing A-5. Example3.cpp

```
#include "stdafx.h"

extern "C" double CalcResult3_(__int64 a, __int64 b, double c, double d);

int _tmain(int argc, _TCHAR* argv[])
{
    __int64 a = 10;
    __int64 b = -15;
    double c = 2.0;
    double d = -3.0;

    double e = CalcResult3_(a, b, c, d);

    printf("a: %lld b: %lld c: %.4lf d: %.4lf\n", a, b, c, d);
    printf("e: %.4lf\n", e);
    return 0;
}
```

Listing A-6. Example3_.asm

```
        .code

; extern "C" double CalcResult3_(int a, int b, double c, double d);

CalcResult3_ proc
        imul rcx,rcx                    ;ecx = a * a
        imul rdx,rdx                    ;edx = b * b

        cvtsi2sd xmm0,rcx               ;convert rcx to DPFP
        cvtsi2sd xmm1,rdx               ;convert rdx to DPFP

        mulsd xmm2,xmm2                 ;xmm2 = c * c
        mulsd xmm3,xmm3                 ;xmm3 = d * d

        addsd xmm0,xmm1                 ;xmm0 = a * a + b * b
        addsd xmm2,xmm3                 ;xmm2 = c * c + d * d
        addsd xmm0,xmm2                 ;xmm0 = sum of squares
        sqrtsd xmm0,xmm0                ;xmm0 = sqrt(sum of squares)
        ret
CalcResult3_ endp
        end
```

Example 4: A Mixed Win32/x64 Project

In this example, you learn how to create a mixed Win32/x64 project that supports both x86-32 and x86-64 assembly language functions.

Create the Project

Use the following steps to create the project:

1. Start Visual Studio.
2. Create a new project named `Example4`. Follow the steps in the Example1 sections entitled "Create the Project" through "Add an Assembly Language File." The source code files should be named `Example4.cpp` and `Example4_.asm`.
3. Select Build | Configuration Manager.
4. In the Configuration Manager dialog box, select **<New...>** under Active Solution Platform (see Figure A-12).
5. In the New Solution Platform dialog box, select **x64** under Type or Select the New Platform (see Figure A-13).
6. Click on OK.
7. Click on Close.

Set the Project Properties

Use the following steps to set up the project's properties:

1. In the Solution Explorer tree control, right-click on `Example4` and select **Properties**.
2. In the Property Pages dialog box, select **All Configurations** for Configuration.
3. Select **Win32** for the Platform.
4. Select Configuration Properties | Microsoft Macro Assembler | Advanced.
5. Set the Use Safe Exception Handlers property to **Yes (/safeseh)**, as shown in Figure A-15.

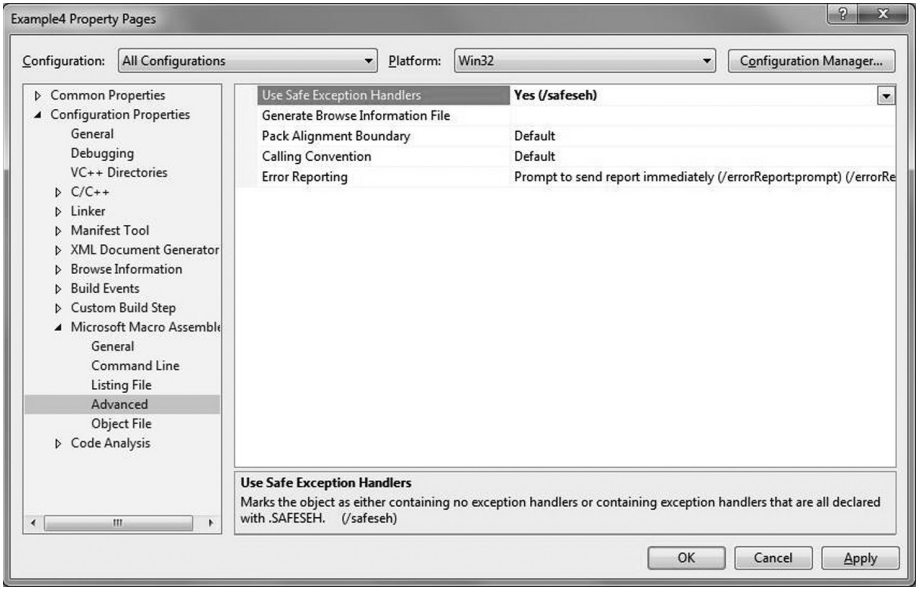

Figure A-15. *Setting the Use Safe Exception Handlers property for the Win32 platform*

6. Select Configuration Properties | Microsoft Macro Assembler | General.
7. Set the Preprocessor Definitions property to `ASMX86_32`, as shown in Figure A-16.

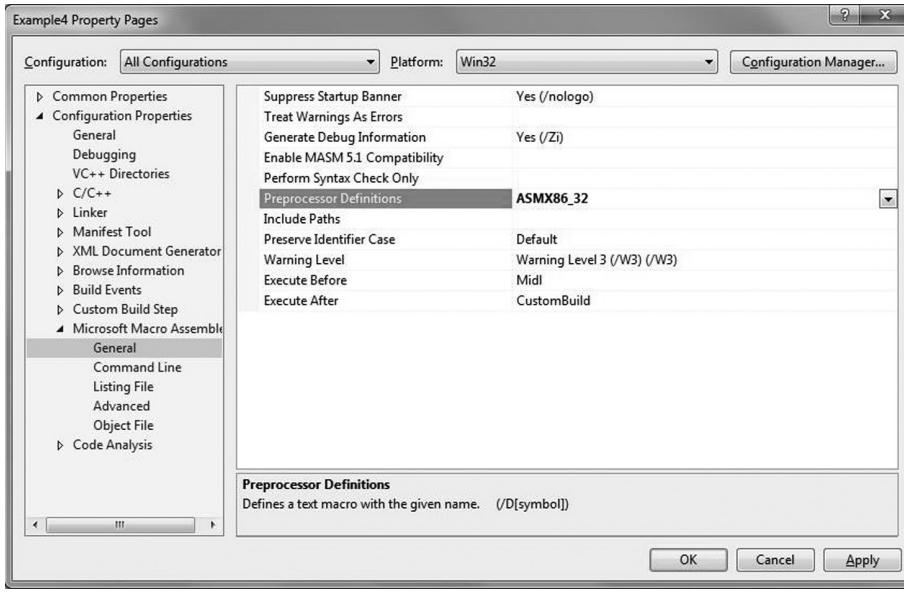

Figure A-16. *Setting the Preprocessor Definitions property for the Win32 platform*

8. Select **x64** for the Platform. Click on Yes if prompted to save any property setting changes.

9. Set the Preprocessor Definitions property to `ASMX86_64`, as shown in Figure A-17.

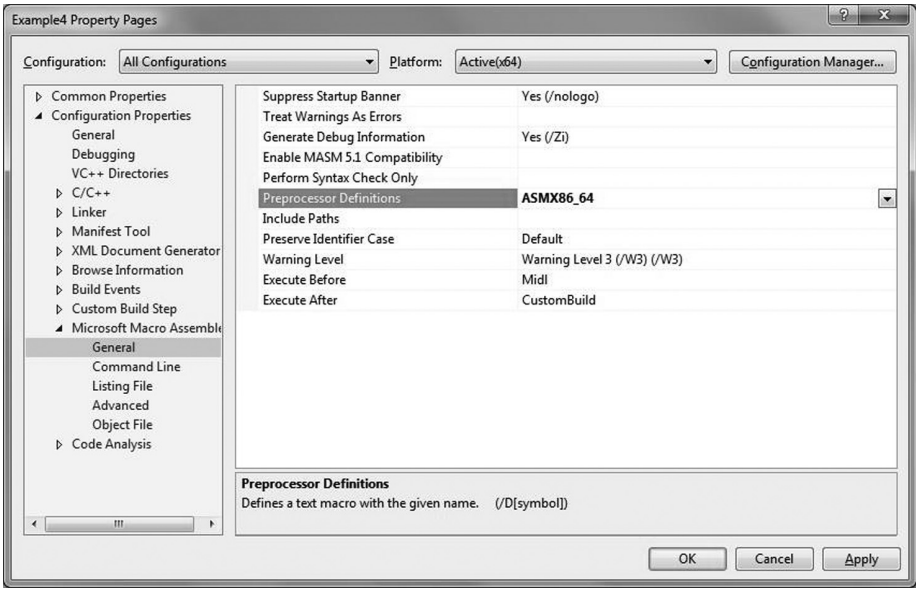

Figure A-17. *Setting the Preprocessor Definitions property for the x64 platform*

10. Click on OK.

Edit the Source Code

Use the following steps to edit the source code:

1. In the Editor window, click on the tab named `Example4.cpp`.
2. Enter the C++ source code shown in Listing A-7.
3. Click on the tab named `Example4_.asm`.
4. Enter the x86 assembly language source code shown in Listing A-8.
5. Select File | Save All (or press Ctrl+Shift+S).

Listing A-7. `Example4.cpp`

```
#include "stdafx.h"

extern "C" bool CalcResult4_(int* y, const int* x, int n);

int _tmain(int argc, _TCHAR* argv[])
{
    const int n = 8;
```

```
    const int x[n] = {3, 2, 5, 7, 8, 13, 20, 25};
    int y[n];

    CalcResult4_(y, x, n);

#ifdef _WIN64
    const char* sp = "x64";
#else
    const char* sp = "Win32";
#endif

    printf("Results for solution platform %s\n\n", sp);
    printf("     x      y\n");
    printf("--------------\n");

    for (int i = 0; i < n; i++)
        printf("%6d %6d\n", x[i], y[i]);
    return 0;
}
```

Listing A-8. Example4_.asm

```
; extern "C" bool CalcResult4_(int* y, const int* x, int n);

IFDEF ASMX86_32
        .model flat,c
        .code

CalcResult4_ proc
        push ebp
        mov ebp,esp
        push ebx
        push esi

        mov ecx,[ebp+8]                 ;ecx = ptr to y
        mov edx,[ebp+12]                ;edx = ptr to x
        mov ebx,[ebp+16]                ;eax = n
        test ebx,ebx                    ;is n <= 0?
        jle Error                       ;jump if n <= 0

        xor esi,esi                     ;i = 0;
@@:     mov eax,[edx+esi*4]             ;eax = x[i]
        imul eax,eax                    ;eax = x[i] * x[i]
        mov [ecx+esi*4],eax             ;save result to y[i]

        add esi,1                       ;i = i + 1
        cmp esi,ebx
        jl @B                           ;jump if i < n
```

```
        mov eax,1                       ;set success return code
        pop esi
        pop ebx
        pop ebp
        ret

Error:  xor eax,eax                     ;set error return code
        pop esi
        pop ebx
        pop ebp
        ret

CalcResult4_ endp
ENDIF

IFDEF ASMX86_64
        .code

CalcResult4_ proc

; Register arguments: rcx = ptr to y, rdx = ptr to x, and r8d = n
        movsxd r8,r8d                   ;sign-extend n to 64 bits
        test r8,r8                      ;is n <= 0?
        jle Error                       ;jump if n <= 0

        xor r9,r9                       ;i = 0;
@@:     mov eax,[rdx+r9*4]              ;eax = x[i]
        imul eax,eax                    ;eax = x[i] * x[i]
        mov [rcx+r9*4],eax              ;save result to y[i]

        add r9,1                        ;i = i + 1
        cmp r9,r8
        jl @B                           ;jump if i < n

        mov eax,1                       ;set success return code
        ret

Error:  xor eax,eax                     ;set error return code
        ret

CalcResult4_ endp
ENDIF
        end
```

Build and Run the Program

Use the following steps to build and run the program:

1. In the Standard toolbar, select **Debug** for the Solution Configuration and **Win32** for the Solution Platform.
2. Press F7 to build the program. If necessary, fix any reported compiler or assembler errors.
3. Press Ctrl+F5 to run `Example4`. Verify that the Console window output matches Figure A-18.

Figure A-18. *Console window output for Example4*

4. In the Standard toolbar, select **x64** for the Solution Platform.
5. Repeat Steps 2 and 3.
6. Select File | Exit (or press Alt+F4) to close Visual Studio.

Additional Include Directories

You can use the following steps to reference additional C++ or MASM include directories in a project:

1. In the Solution Explorer tree control, right-click on the project name and select **Properties**.
2. In the Property Pages dialog box, select **All Configurations** for Configuration.

3. If the project includes both Win32 and x64 platforms, select **All Platforms** for Platform.

4. Select Configuration Properties | C/C++ | General.

5. Set the Additional Include Directories property to the required directories, as shown in Figure A-19. Use a semicolon to separate multiple directories.

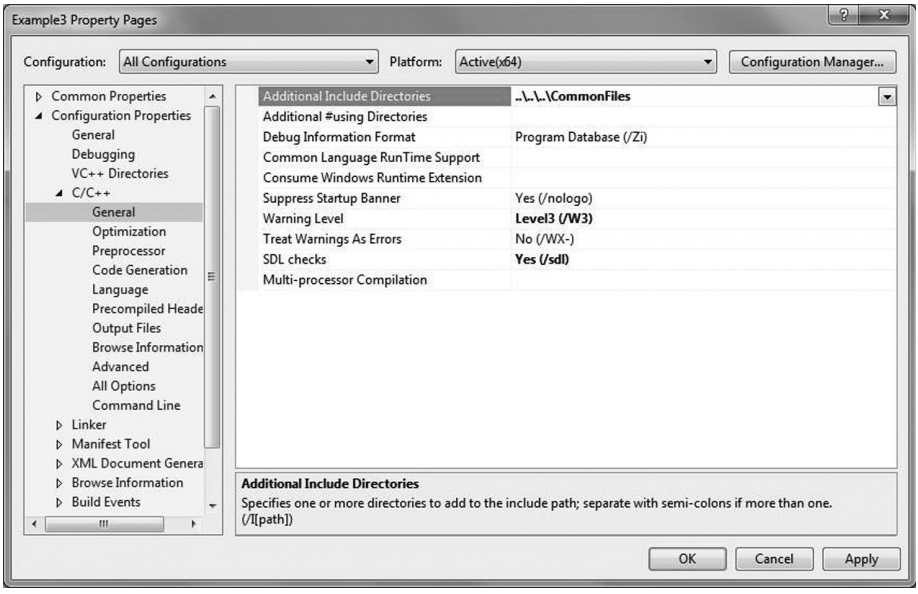

Figure A-19. *Properties page for additional C++ include directories*

6. Click on OK.

You can use the following steps to reference additional MASM include directories in a project:

1. In the Solution Explorer tree control, right-click on the project name and select **Properties**.

2. In the Property Pages dialog box, select **All Configurations** for Configuration.

3. If the project includes both Win32 and x64 platforms, select **All Platforms** for Platform.

4. Select Configuration Properties | Microsoft Macro Assembler | General.

5. Enter the MASM include directories in the property labeled Include Paths, as shown in Figure A-20. Use a semicolon to separate multiple directories.

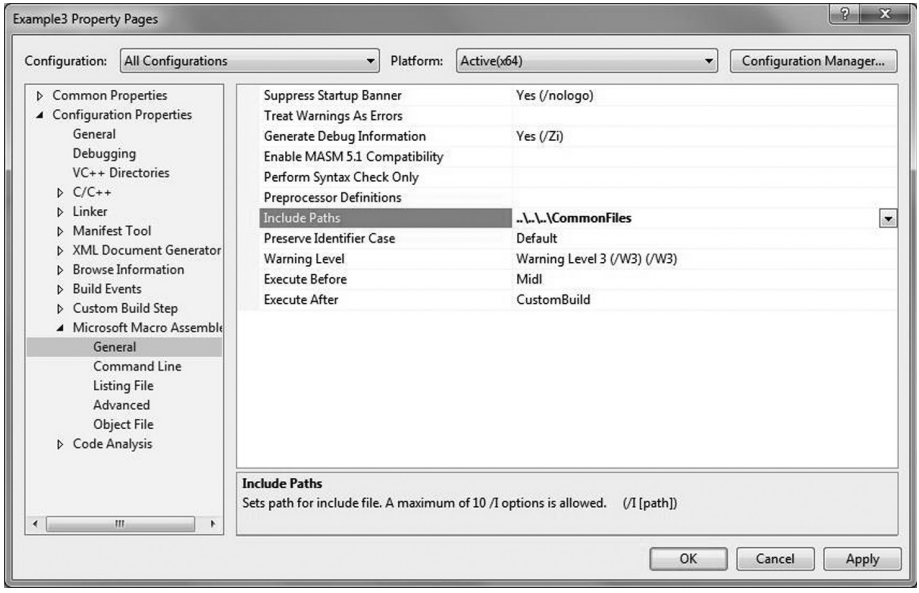

Figure A-20. *Properties page for the MASM include directories*

6. Click on OK.