Cycle Model Studio

Version 9.0.0

Windows Visual C++ SystemC Integration Application Note

Non-Confidential

Cycle Model Studio
Windows Visual C++ SystemC Integration Application Note

Release Information

The following changes have been made to this document.

Change History

Date	Issue	Confidentiality	Change
November 2016	A	Non-Confidential	Restamp release with 9.0.0

Abstract

This application note describes how to integrate a Cycle Model with the Microsoft Visual C++ .NET Windows environment and Accellera SystemC. This process includes setting up the Cycle Model Studio Windows environment, compiling the Cycle Model on Linux, setting up the SystemC Windows environment, and integrating with Visual C++.

Integration Process

The basic integration procedure includes the following steps:

1. Setting up the Cycle Model Studio environment on Windows.
2. Setting up the SystemC environment on Windows.
3. Cross compiling the Windows Cycle Model on Linux.
4. Integrating with Visual C++.

Note that this procedure has been verified on Windows XP.

Before you begin, make sure that your Cycle Model Studio environment is set up correctly on Linux, and that you can compile a Cycle Model for use on Linux.

Step 1 – Cycle Model Studio Windows Environment Setup

First, you must have Read access to an ARM Cycle Model Studio release directory on Windows. You can mount the Cycle Model Studio release directory on to your PC, copy the required directories/files to your PC, or install Cycle Model Studio software on your Windows machine (see the *Cycle Model Studio Installation Guide*). If you mount or copy the release data from Linux, make sure the release installation included the Windows data. If you install the Cycle Model Studio Windows release, make sure it matches the release version installed on Linux.

At a minimum, you need the following directories from CARBON_HOME:

- include
- lib
- makefiles
- Win
- systemc (may be needed if SystemC is not already installed)

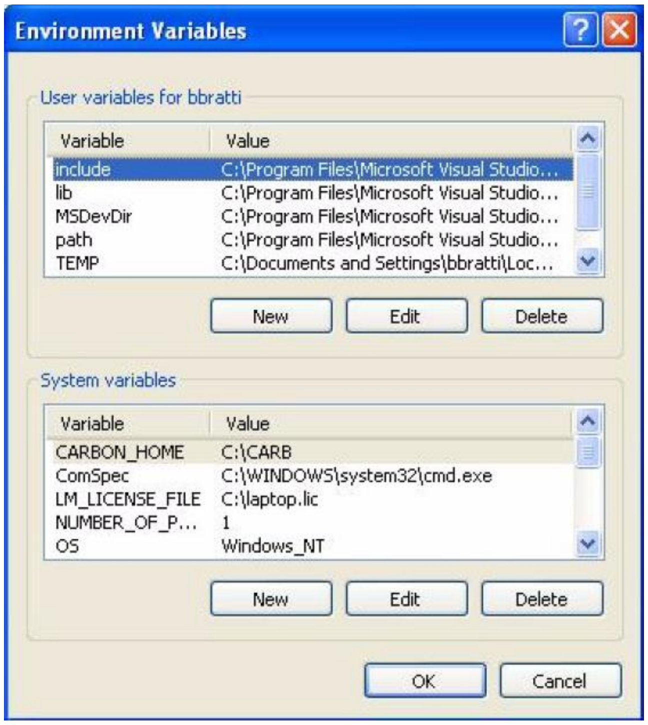
In addition, you must set the following environment variables appropriately (see the *Cycle Model Studio Installation Guide* for additional details):

- CARBON_HOME
- ARMLMD_LICENSE_FILE (or LM_LICENSE_FILE)

Set these environment variables in Windows by selecting:

Start > **Control Panel** > **System** > **Advanced** > **Environment Variables**

The following dialog displays.



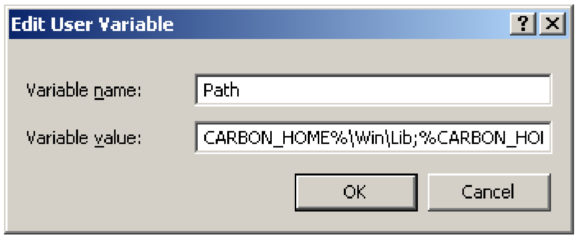
Variable	Value
include	C:\Program Files\Microsoft Visual Studio...
lib	C:\Program Files\Microsoft Visual Studio...
MSDevDir	C:\Program Files\Microsoft Visual Studio...
path	C:\Program Files\Microsoft Visual Studio...
TEMP	C:\Documents and Settings\bbratti\Loc...

Variable	Value
CARBON_HOME	C:\CARB
ComSpec	C:\WINDOWS\system32\cmd.exe
LM_LICENSE_FILE	C:\laptop.lic
NUMBER_OF_P...	1
OS	Windows_NT

Lastly, add the following to your Windows PATH environment variable, based on your CARBON_HOME variable.

%CARBON_HOME%\Win\lib; %CARBON_HOME%\Win\lib\winx\shared

Depending on your environment, you may need to explicitly expand %CARBON_HOME%.



Variable name: Path

Variable value: CARBON_HOME%\Win\Lib;%CARBON_HOI

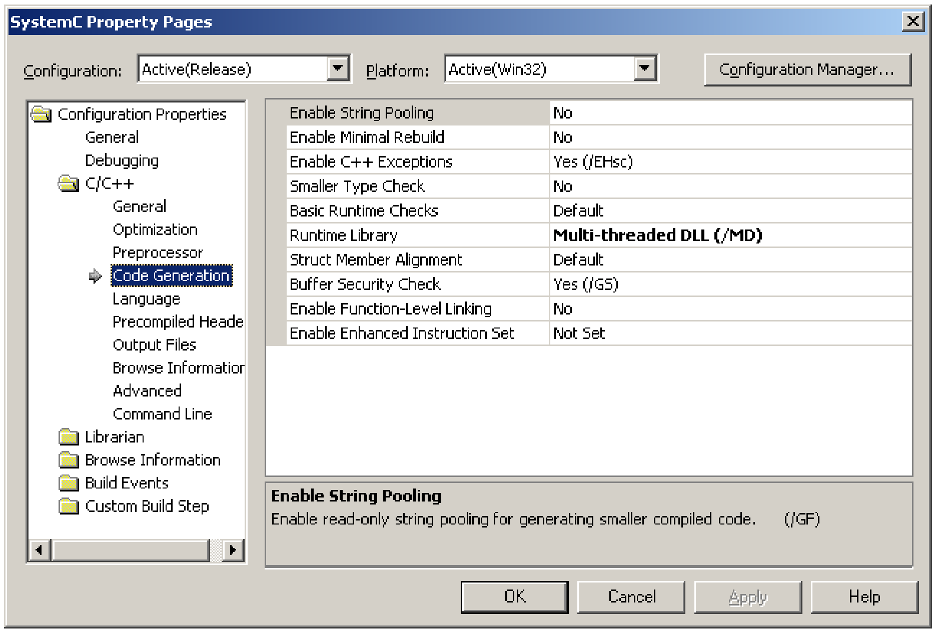
Note: For UNIX users, on Windows, the %VARIABLE_NAME% notation is identical to shell variable accessing via $VARIABLE_NAME.

Step 2 – SystemC Windows Environment Setup

The SystemC 2.2 distribution file is available at %CARBON_HOME%\systemc\systemc-dist.tgz. Unpack the “systemc-dist.tgz” with WinZip or other tool at an appropriate location (for example, C:\Program Files\SystemC\SystemC_2_2). Create a Windows environment variable called SYSTEMC_HOME that points at the SystemC installation directory. Open the %SYSTEMC_HOME%\INSTALL file with a text editor for details on SystemC installation instructions.

Go to %SYSTEMC_HOME%\msvc71\SystemC, and open the SystemC Project file, "SystemC.vcproj" with Visual Studio. This can be accomplished by double clicking on "SystemC.vcproj".

1. After opening the project, modify the project to use the /MD option for the Release configuration, and the /MDd option for the Debug configuration (rather than the /ML (or /MLd) compiler options) so that it uses the multi-threaded DLL version of the C runtime libraries.



Enable String Pooling	No
Enable Minimal Rebuild	No
Enable C++ Exceptions	Yes (/EHsc)
Smaller Type Check	No
Basic Runtime Checks	Default
Runtime Library	**Multi-threaded DLL (/MD)**
Struct Member Alignment	Default
Buffer Security Check	Yes (/GS)
Enable Function-Level Linking	No
Enable Enhanced Instruction Set	Not Set

Enable String Pooling
Enable read-only string pooling for generating smaller compiled code. (/GF)

- From the Project menu, choose Properties. Choose "C/C++" -> "Code Generation" and set the "Runtime Library" field to "**Multi-threaded Debug DLL (/MDd)**" for the Configuration: *Debug*, and to "**Multi-threaded DLL (/MD)**" for the Configuration: *Release*.

2. Build debug and release configurations:

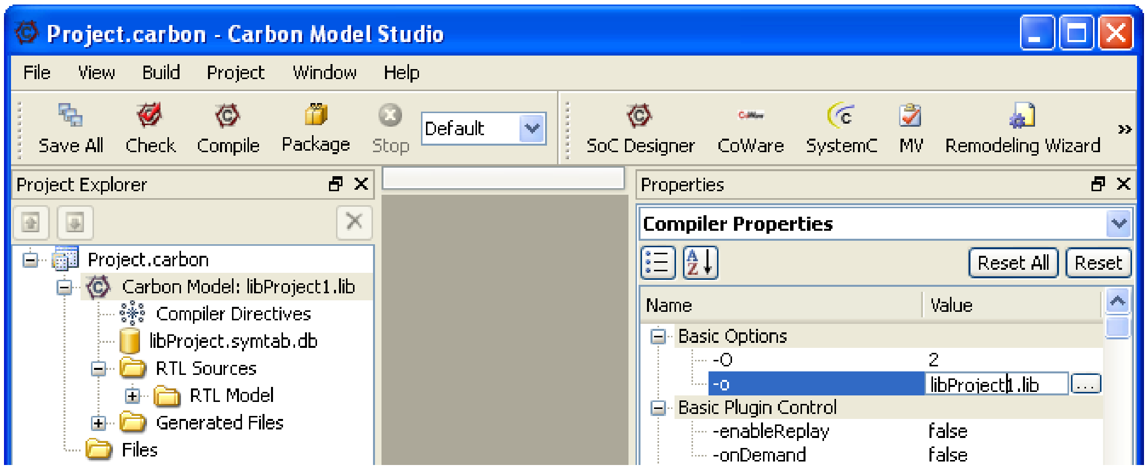
- To build the Release configuration:
  i. Set "Solution Configuration" to "Release".
  ii. Select Build -> Build Solution.

- To build Debug configuration:
  i. Set "Solution Configuration" to "Debug".
  ii. Select Build -> Build Solution.

Step 3 – Creating the Cycle Model

To compile a Cycle Model for Windows on a Linux machine, specify the *.lib* file extension for the Cycle Model name (in the format *lib<design>.lib*) using the `-o` Cycle Model Compiler option. Note that you need to replace *<design>* with the name of the design.

- Using Cycle Model Studio, specify the file name in the Basic Options section of the Compiler Properties, as shown below:


Project.carbon - Carbon Model Studio
File View Build Project Window Help
Save All Check Compile Package Stop Default
SoC Designer CoWare SystemC MV Remodeling Wizard
Project Explorer
Project.carbon
Carbon Model: libProject1.lib
Compiler Directives
libProject.symtab.db
RTL Sources
RTL Model
Generated Files
Files
Properties
Compiler Properties
Reset All Reset

Name	Value
Basic Options	
-O	2
-o	libProject1.lib
Basic Plugin Control	
-enableReplay	false
-onDemand	false

- Using the Cycle Model Compiler (cbuild) command line, enter:

```
cbuild -o lib<design>.lib ...
```

  instead of `-o lib<design>.a`, which generates a Cycle Model for UNIX.

The Cycle Model Studio or the Cycle Model compiler outputs a number of files, including the following, which are required for compiling/running the executable on Windows:

- *lib<design>.lib* – Windows static object library
- *lib<design>.h* – include file for interfacing to the Cycle Model
- *lib<design>.symtab.db* – Cycle Model database with information about *all* internal signals
- *lib<design>.io.db* – Cycle Model database with only a subset of the signals—only the IOs and additional signals marked as observable or depositable

Additionally, for use with SystemC, the following files generated by the Cycle Model Studio (or by **systemCWrapper**) are needed:

- *lib<design>.systemc.cpp* –SystemC wrapper file
- *lib<design>.systemc.h* –SystemC header file

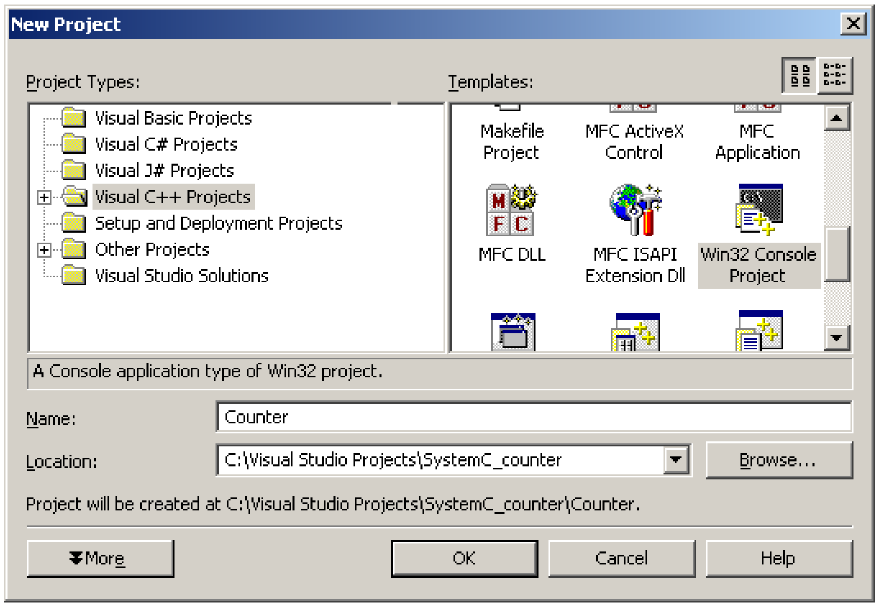
All files listed above must be copied to (or be accessible to) your Windows environment.

Step 4 – Visual C++ Integration

To integrate the Cycle Model, you must either create a new Visual C++ project or make changes to your existing project. The following are the steps to open and set up a new project:

1. Start Visual C++ .Net 2003 and open a new Win32 Console Project.
    - Select File -> New Project and create a project with the following settings:

        - Project Types: Visual C++ Projects
        - Name: <Enter Project Name>
        - Location: <Enter Project Dir Path>
        - Template: Win32 Console Project
    - Click **OK**.
    - Select the "Application Settings" page of the "Win32 Application Wizard" and make sure the 'Empty project' box is selected.
    - Then select "Finished".



New Project

Project Types: Visual C++ Projects

Templates: Win32 Console Project

A Console application type of Win32 project.

Name: Counter

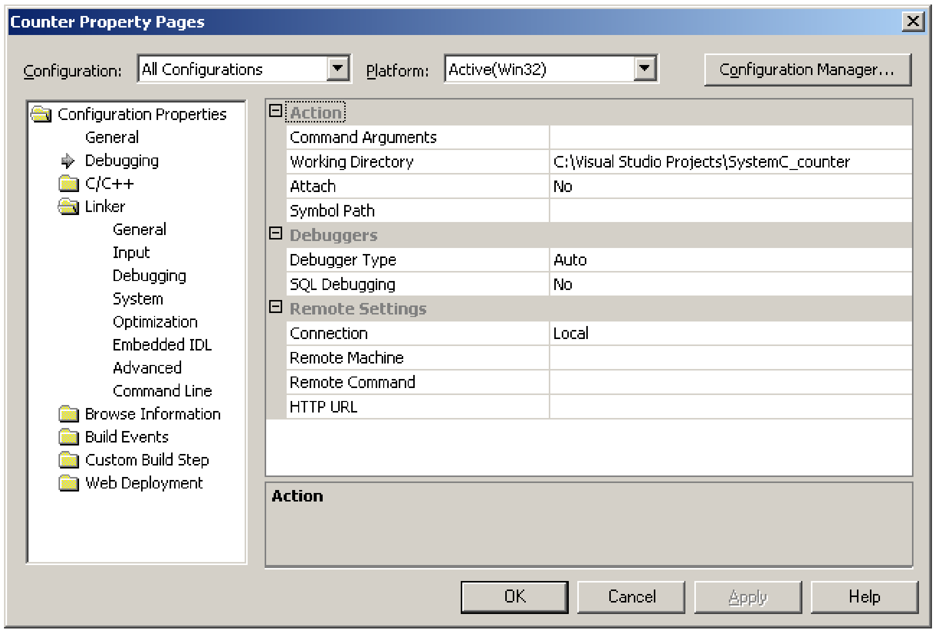
Location: C:\Visual Studio Projects\SystemC_counter

Project will be created at C:\Visual Studio Projects\SystemC_counter\Counter.

2. Add C++/SystemC files to the project, etc. This can be accomplished by dragging and dropping the files into the Visual Studio "Solution Explorer". For the Cycle Model, the following files must be added (including the SystemC wrapper files):
    - Source Files: *lib<design>.systemc.cpp*
    - Header Files: *lib<design>.h*, *lib<design>.systemc.h*

3. Open the Project Property Page (Project -> Properties), and set the following properties:
    - Set "Configuration:" to "All Configurations".
    - From the "Configuration Properties" tab, select the "Debugging" properties and set "Working Directory" to the directory containing the Cycle Model DB file(s) (*lib<design>.symtab.db* and/or *lib<design>.io.db*).

Note: The default location for the Cycle Model DB file is the current directory. The DB file is embedded in the Cycle Model by default when it is compiled (see the *ARM Cycle Model Compiler User Manual*, ARM DUI 0957, for information on the compiler options -`embedIODB` and -`noFullDB`). Additionally, the `carbonSetFilePath()` C-API function can be used in the Cycle Model Wrapper to specify other locations to find the DB file.



Action	
Command Arguments	
Working Directory	C:\Visual Studio Projects\SystemC_counter
Attach	No
Symbol Path	
Debuggers	
Debugger Type	Auto
SQL Debugging	No
Remote Settings	
Connection	Local
Remote Machine	
Remote Command	
HTTP URL	

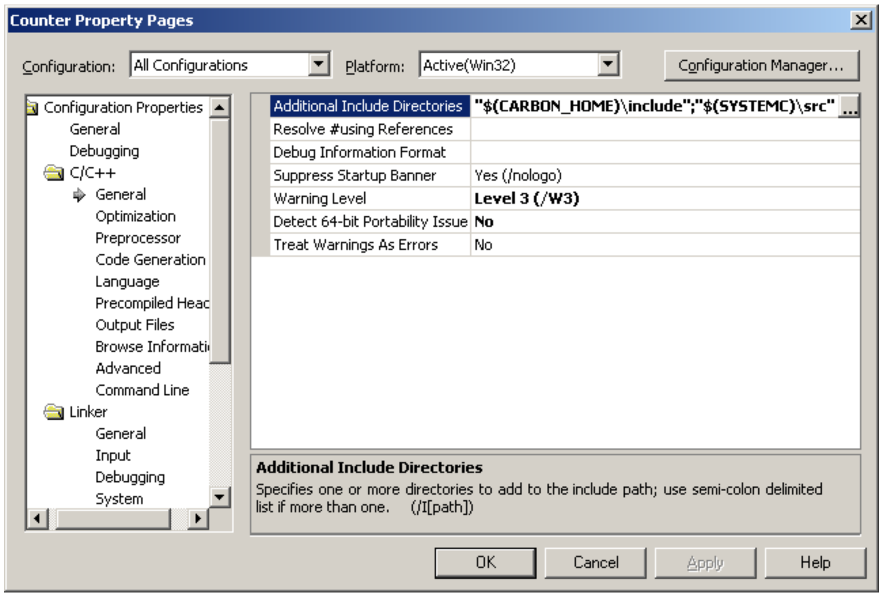
Action

- From the "C/C++" tab,
    i. Select the "General" properties and set "Additional Include Directories" to:
    "$(CARBON_HOME)\include";"$(SYSTEMC_HOME)\src".

    Note: "$(SYSTEMC_HOME)\src" is required by SystemC.

    ii. Set "Detect 64-bit Portability Issues" to:
    No

    Note: This is required by SystemC.



Additional Include Directories	"$(CARBON_HOME)\include";"$(SYSTEMC)\src"
Resolve #using References	
Debug Information Format	
Suppress Startup Banner	Yes (/nologo)
Warning Level	**Level 3 (/W3)**
Detect 64-bit Portability Issue	**No**
Treat Warnings As Errors	No

Additional Include Directories
Specifies one or more directories to add to the include path; use semi-colon delimited list if more than one. (/I[path])

iii. Select the "Language" properties and set "Enable Run-Time Type Info" to:
   Yes (/GR)

   Note: This is required by SystemC.

iv. Select the "Command Line" properties and in the "Additional Options:" box add:
   /vmg

   Note: This is required by SystemC. Failure to do this can cause your program to crash.

- From the Linker tab, select the "Input" properties and in the "Additional Dependencies" box add:
  lib<design>.lib libcarbon5.lib ffwAPIexp.lib SystemC.lib libgcc.lib

  Note: "SystemC.lib" is required by SystemC.

- Click **Apply**.

- Set "Configuration:" to "Release".
  - From the "C/C++" tab, select the "Code Generation" properties and set "Runtime Library" to "Multi-threaded DLL (/MD)".
  - From the Linker tab, select the "General" properties and in the "Additional Library Directories" box add:
    "$(SYSTEMC_HOME)\msvc71\SystemC\Release";"$(CARBON_HOME)\Win\lib";"$(CARBON_HOME)\Win\Lib\winx\shared";"$(CARBON_HOME)\Win\winx\lib\gcc\i386-mingw32msvc\3.4.5-a";<*directory containing the Cycle Model, the libdesign.lib file*>
  - Click **Apply**.

- Set "Configuration:" to "Debug".
  - From the "C/C++" tab, select the "Code Generation" properties and set "Runtime Library" to "Multi-threaded Debug DLL (/MDd)".
  - From the Linker tab, select the "General" properties and in the "Additional Library Directories" box add:
    "$(SYSTEMC_HOME)\msvc71\SystemC\Debug";"$(CARBON_HOME)\Win\lib";"$(CARBON_HOME)\Win\Lib\winx\shared";"$(CARBON_HOME)\Win\winx\lib\gcc\i386-mingw32msvc\3.4.5-a";<*directory containing the Cycle Model, the libdesign.lib file*>
- Click **OK**.

4. Build the Project.
   - Set the configuration to either "Debug" or "Release".

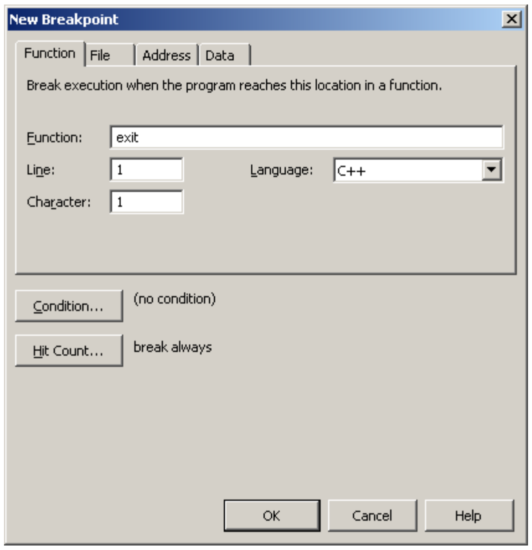
- Select Build -> Build Solution.

5. Run the Simulation.
   - Set the configuration to either “Debug” or “Release”.
   - Select Debug -> Start.

6. Debug Simulation Startup Issues.

   If the simulation reports an exception error at the beginning, there may be a problem finding the ARM license or finding the Cycle Model database file. To determine the cause, perform the following steps:

   - Set a breakpoint on the exit function:
     - Open the New Breakpoint window by selecting Debug -> New Breakpoint (or Ctrl+B).
     - In the Function field, type “exit”.
     - Click **OK**.



Break execution when the program reaches this location in a function.

Function: exit
Line: 1
Language: C++
Character: 1
(no condition)
break always

   - Start the simulation by selecting Debug -> Start (or F5).
   - Examine the error message(s) displayed in the Command Prompt window that was open when the simulation started.

   If it is a license issue, you may see the following error message:

```
Error: Carbon: License checkout failed - 'Checkout of crbn_vsp
failed: No VENDOR_STRING match for SIG=Carbon - checkout failed:
```

   In this case, examine the ARMLMD_LICENSE_FILE environment variable to verify that it is set to a valid license.

   If it cannot find the Cycle Model database file, you may see the following error message:

```
Error: Carbon: Error opening file libcounter.symtab.db: No such file
or directory or path is not readable. Current dir: C:\Visual Studio
Projects\counter -- Search path: ./
In file: c:\visual studio projects\counter\libcounter.systemc.h:295
```

In this case, verify that the "Working Directory" (**Projects -> Property -> Configuration Properties -> Debugging -> Working Directory**) is pointing to the directory containing the Cycle Model database file.

7. Configuration for Simulation Portability.

   When the simulation starts, the Cycle Model loads certain DLLs. If the simulation will be run on a machine without access to the Cycle Model Studio installation, you will need to copy those DLLs to one of the following locations: in the same directory as the .exe file that runs the simulation, in the process's current directory, or in a directory listed in the PATH environment variable. The DLLs that are affected are:

   %CARBON_HOME%\Win\lib\ffwAPI.dll
   %CARBON_HOME%\Win\lib\winx\shared\libcarbon5.dll
   %CARBON_HOME%\Win\lib\winx\shared\libcarbonaux.dll