

Configuring Microsoft® Visual Studio Projects to Support the AMD64 Architecture

July 21, 2004

Abstract

This paper provides information about converting Microsoft® Visual Studio projects to support the AMD64 architecture. It provides guidance for developers on how to configure their existing development environment and project settings of current 32-bit applications for Microsoft Windows to target 64-bit Microsoft Windows on AMD64 processors. It gives instructions and tips on how to configure Microsoft Visual Studio 6 and Microsoft Visual Studio .Net environments to work with the AMD64 command-line tools and on how to modify their project settings for AMD64 processors.

Contents

Introduction	2
Microsoft Tools for the AMD64 Architecture	2
Microsoft Visual Studio.....	2
Microsoft Visual Studio 6	3
Microsoft Visual Studio .Net.....	5
Microsoft Visual Studio “Whidbey”	6
Debugging AMD64 Applications.....	8
Call to Action	10
Resources and References	11
About the Author and Acknowledgments.....	12



Windows Hardware Engineering Conference

© 2004 Advanced Micro Devices, Inc.

All rights reserved.

The contents of this document are provided in connection with Advanced Micro Devices, Inc. (“AMD”) products and technology. AMD makes no representations or warranties with respect to the accuracy or completeness of the contents of this publication and reserves the right to make changes to specifications and product and technology descriptions at any time without notice. No license, whether express, implied, arising by estoppel or otherwise, to any intellectual property rights is granted by this publication. AMD assumes no liability whatsoever, and disclaims any express or implied warranty, relating to its products including, but not limited to, the implied warranty of merchantability, fitness for a particular purpose, or infringement of any intellectual property right.

WinHEC Sponsors’ Disclaimer: The contents of this document have not been authored or confirmed by Microsoft or the WinHEC conference co-sponsors (hereinafter “WinHEC Sponsors”). Accordingly, the information contained in this document does not necessarily represent the views of the WinHEC Sponsors and the WinHEC Sponsors cannot make any representation concerning its accuracy. THE WINHEC SPONSORS MAKE NO WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS INFORMATION.

AMD, the AMD Arrow logo, AMD Opteron, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Microsoft and Windows are trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries. Other product and company names mentioned herein may be the trademarks of their respective owners.

Introduction

Now that 64-bit computing is available in desktop systems and even in laptops with AMD64 processors, it brings the opportunity to run computational applications that were never possible before other than on “big iron” systems. Even though Microsoft Windows with 64-bit Extensions can run 32-bit applications at full speed on AMD64 processors, many applications can benefit from running natively on such systems. Porting such applications to the AMD64 architecture is in most cases simple and Microsoft will soon make available to developers Microsoft Visual Studio “Whidbey” which will fully support the AMD64 architecture. However, today it is already possible to use older versions of Microsoft Visual Studio along with a Platform SDK that supports the AMD64 architecture.

Microsoft Tools for the AMD64 Architecture

As of the writing of this paper, the current Platform SDK (see <http://www.microsoft.com/msdownload/platformsdk/sdkupdate>) comes with the Microsoft C/C++ compiler that supports the AMD64 architecture. Along with the compiler itself, obligatory tools such as a linker and a librarian with full support for the AMD64 architecture are also provided.

Moreover, for low-level code, a version of MASM supporting the AMD64 instruction set is also added to the tool set. However, because the Applications Binary Interface (ABI) is totally new for the AMD64 architecture, assembler code has to be rewritten to take advantage of 64-bit features. Instead, it is recommended that assembler code be converted to high-level code using the compiler’s intrinsics, with the advantage that it can be compatible for both AMD64 and Win32 platforms.

The 64-bit Microsoft Windows for AMD64 processors comes with NTSD, a symbolic debugger that fully supports debugging of both 32-bit applications and applications for the AMD64 architecture. In addition to NTSD, Microsoft Windbg also supports debugging of 32-bit applications under 64-bit Microsoft Windows for AMD64 processors, as well remote debugging of 64-bit drivers. Microsoft has also made available a beta version of Microsoft Windbg that supports native debugging of both 32-bit applications and drivers and applications for the AMD64 architecture under 64-bit Microsoft Windows for AMD64 processors (see <http://www.microsoft.com/whdc/ddk/debugging/installAMDbeta.mspx>).

Besides the tools, the header files and libraries for Microsoft Windows for AMD64 processors complete the development environment. Those who feel comfortable with using command-line tools will be able to start working with the Microsoft tools for the AMD64 architecture immediately.

Microsoft Visual Studio

Microsoft Visual Studio 6 and .Net remain popular and can be easily tailored to use the tool set for the AMD64 architecture from the Platform SDK. The next-generation of Microsoft Visual Studio, codenamed “Whidbey”, provides its own toolset supporting the AMD64 architecture and allows for easy targeting of applications for multiple platforms in the same project. Microsoft Visual Studio “Whidbey” can debug 32-bit applications and drivers locally and 64-bit applications and drivers remotely. Other versions of Microsoft Visual Studio require using Microsoft Windbg to debug 64-bit applications and drivers.

Microsoft Visual Studio 6

Microsoft Visual Studio 6 is configured at setup to use its own toolset. One can check this configuration by selecting the menu options “Tools | Options” and then looking at the “Directories” tab, where the paths to the tools, header files and libraries are specified (see Figure 1).

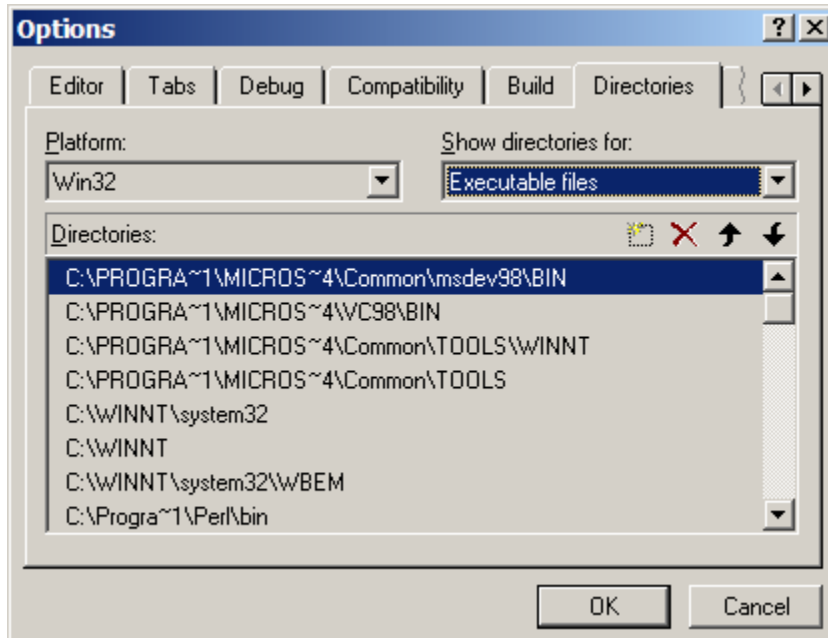


Figure 1: Microsoft Visual Studio 6 directories configuration.

However, configuring the environment through this dialog box is quite laborious and makes it very difficult to target the AMD64 platform at one time and the Win32 platform at another.

Instead, one can use a command-line option for Microsoft Visual Studio 6 that captures the contents of related environment variables into its default configuration, i.e., the variables `PATH`, `INCLUDE` and `LIB`. This option, `/USEENV`, can then be used after evoking the batch file in the PSDK that sets up these environment variables. For example:

```
call "C:\Program Files\SDK\SetEnv.Bat" /AMD64 /RETAIL
start "" "C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin\MSDEV.EXE" /useenv
```

As these settings are persistent between sessions, it is necessary to perform similar steps to restore the settings for the Win32 platform:

```
call "C:\Program Files\Microsoft Visual Studio\VC98\Bin\VCVARS32.BAT"
start "" "C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin\MSDEV.EXE" /useenv
```

Make sure that if the environment configuration was changed to add the paths to special development kits they are also added to the environment variables before invoking the environment for both the AMD64 and the Win32 platforms. For example, if DirectX is used in a project:

```
call "C:\Program Files\SDK\SetEnv.Bat" /AMD64 /RETAIL
set INCLUDE=C:\Program Files\DDK\DX9\Include;%INCLUDE%
set LIB=C:\Program Files\DDK\DX9\Lib\AMD64;%LIB%
start "" "C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin\MSDEV.EXE" /useenv
```

Because the compiler from the Platform SDK may be newer than that in Microsoft Visual Studio 6, it may implement more current C and C++ languages standards. If the developer wants to take advantage of them in applications by using the 32-bit compiler in the Platform SDK also for the Win32 platform, a similar procedure to that for the AMD64 platform can be used:

```
call "C:\Program Files\SDK\SetEnv.Bat" /RETAIL
start "" "C:\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin\MSDEV.EXE" /useenv
```

It is a good idea to create batch files with these procedures for easily switching between targets.

Once the environment is properly configured to use the tools for the AMD64 architecture, it is necessary to create the project configurations for the AMD64 platform by selecting the menu options "Build | Configurations" and then clicking on the "Add" button. In the "Configuration" text field, add a name, for example "Release AMD64", and choose from the list "Copy settings from" the corresponding configuration for the Win32 platform, for example "Win32 Release". Repeat this step for the debug configuration and any other configuration in the project, if any, so that for each Win32 configuration there is a corresponding one for the AMD64 platform (see Figure 2).

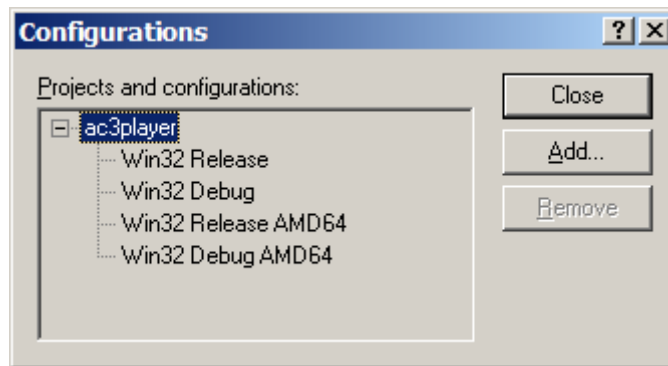


Figure 2: Microsoft Visual Studio 6 project configurations.

Note that because Microsoft Visual Studio 6 can target natively only Win32, it prefixes the configuration names with "Win32", but it can be ignored.

A few options need to be manually set in the configurations targeting the AMD64 platform. For each sub-project, select the menu options "Project | Settings":

- In all configurations for the AMD64 platform, in the "General" tab, make sure to change the output directories in order to avoid confusing the environment by potentially mixing AMD64 and Win32 object files.
- In all configurations for the AMD64 platform, in the "Link" tab, make sure to append `/machine:AMD64` to the options list. The environment will not allow the option `/machine:I386` to be removed, but as long as `/machine:AMD64` comes after it, everything is fine.

- In all configurations for the AMD64 platform, in the “C/C++” tab, in the “General” category, make sure that if any debug info is enabled, it is not “Program Database for Edit and Continue”.
- In the debug configuration the AMD64 platform, in the “C/C++” tab, it is a good idea to add the option `/Wp64` to the options list to help to make the source code 64-bit compliant.

The compiler for the AMD64 architecture may not support some options supported by the compiler that comes with Microsoft Visual Studio 6. Such options will generate appropriate warning messages and it is better to remove them from the configurations for the AMD64 platform where they occur.

At this point, the developer should select “Build | Rebuild All” to start porting the application.

Microsoft Visual Studio .Net

As with Microsoft Visual Studio 6, Microsoft Visual Studio .Net is setup to use its own toolset. It can be checked by selecting the menu options “Tools | Options” and selecting branch “Projects | VC++ Directories” (see Figure 3).

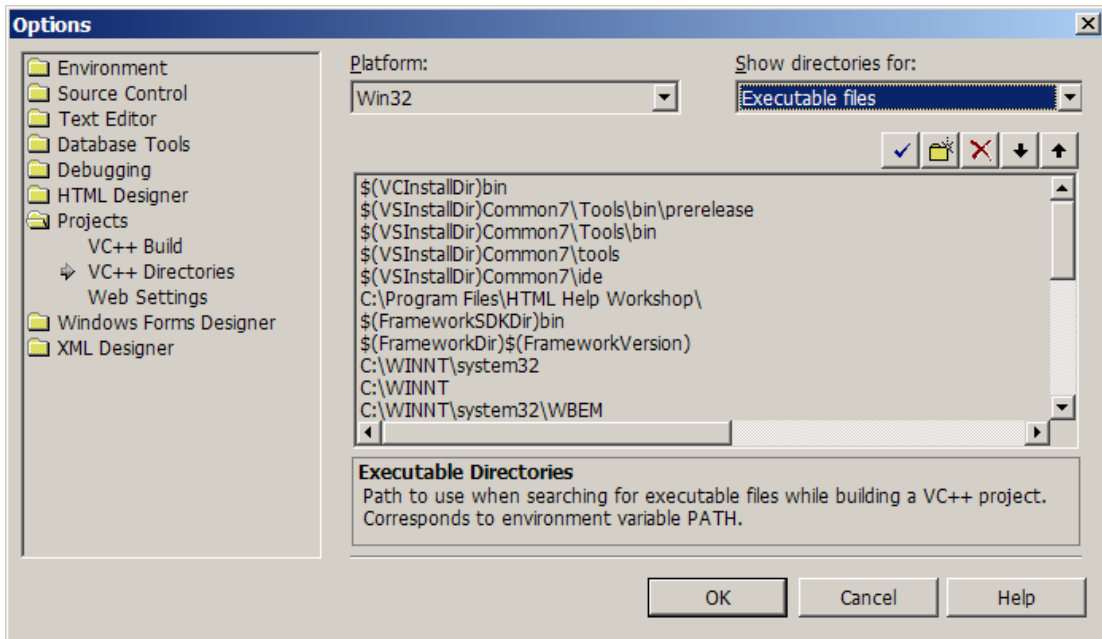


Figure 3: Microsoft Visual Studio .Net directories configuration.

As in its previous generation, Microsoft Visual Studio .Net can also be easily reconfigured through environment variables by using the command-line option `/USEENV` when starting the environment.

The procedure to create project configurations for the AMD64 platform is also similar. Select the menu option “Build | Configuration Manager” and then “<New...>” in “Active Solution Configuration”. Type the name of the configuration in “Solution Configuration Name”, for example “Release AMD64”, choose from the list “Copy Settings from” the corresponding configuration for Win32, for example “Release”, and uncheck “Also create new project configuration(s)” (see Figure 4). Repeat this step for debug configurations and others, if any.

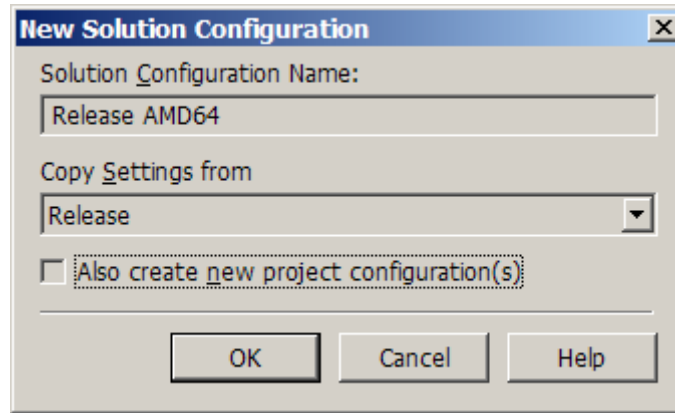


Figure 4: Adding new configuration to Microsoft Visual Studio .Net.

Again, some options must be manually changed to make sure that the newly created configuration will work with the tools for the AMD64 architecture. Navigate each sub-project that make up the solution and edit select “Properties” in the menu brought up by a right-click:

- In all configurations for the AMD64 platform, in the “General” branch, make sure to change the output directories in order to avoid confusing the environment by potentially mixing AMD64 and Win32 object files.
- In all configurations for the AMD64 platform, in the “Linker | Command Line” branch, type `/machine:AMD64` in the “Additional Options” box, overwriting any other instance of this option if any existed.
- In all configurations for the AMD64 platform, in the “C/C++ | General” branch, make sure that if the “Debug Information Format” is enabled, it is not “Program Database for Edit and Continue”.
- In the debug configuration for the AMD64 platform, in the “C/C++ | General” branch, it is a good idea to add the option `/Wp64` to the options list to help to make the source code 64-bit compliant.

The environment configuration is now complete and all the modules should be rebuilt. Select the menu options “Build | Rebuild Solution” to start porting the application to the AMD64 architecture.

Microsoft Visual Studio “Whidbey”

Microsoft Visual Studio “Whidbey” is a preview of the next-generation development environment. It builds on Microsoft Visual Studio .Net and adds support for multi-platform projects.

Unlike the previous versions of Microsoft Visual Studio, it is configured at setup with toolsets for all the platforms it supports, e.g. the Win32 and AMD64 platforms (see Figure 5).

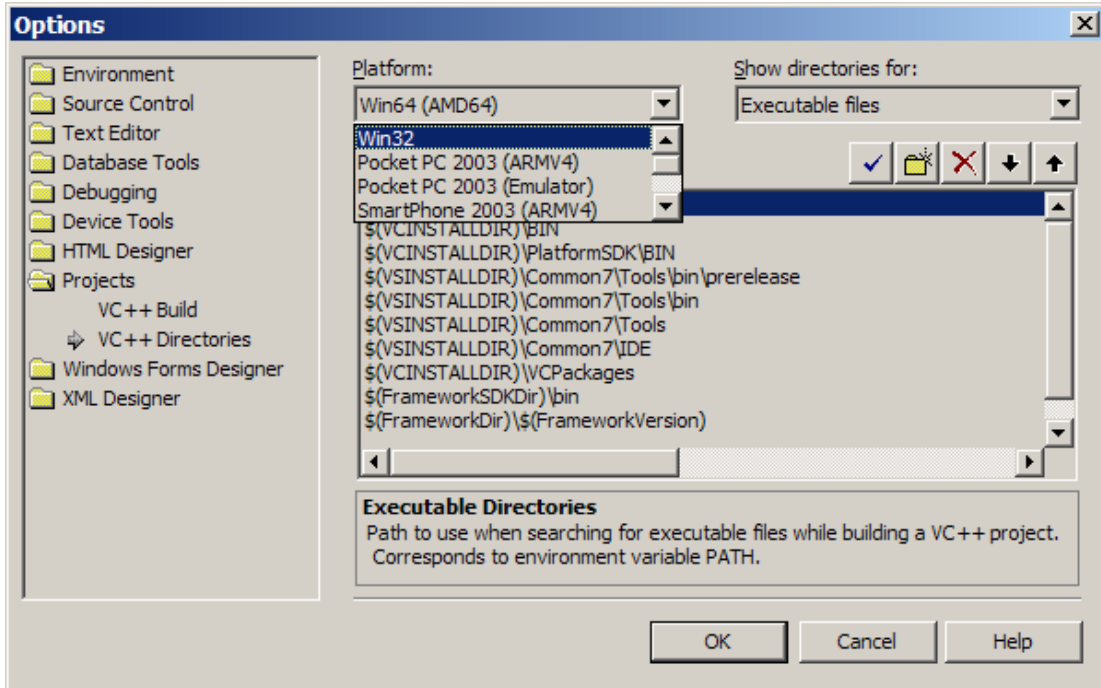


Figure 5: Microsoft Visual Studio “Whidbey” directories configuration.

When Microsoft Visual Studio “Whidbey” opens older versions of the project files, it first converts them to its format version. Once this conversion is complete, creating configurations to support the AMD64 platform now involves two steps: one to create the configuration itself and another to create the target platform.

Select the menu options “Build | Configuration Manager” and select “<New...>” from the “Platform” list and choose “Win64 (AMD64)” in “Select New Platform” and “Win32” in “Copy Settings from”; do check “Also create new solution”. This creates a configuration for the AMD64 platform for each existing one for the Win32 platform. Without closing the Configuration Manager window, create a new AMD64 platform for each sub-project, however unchecking “Also create new solution” this time.

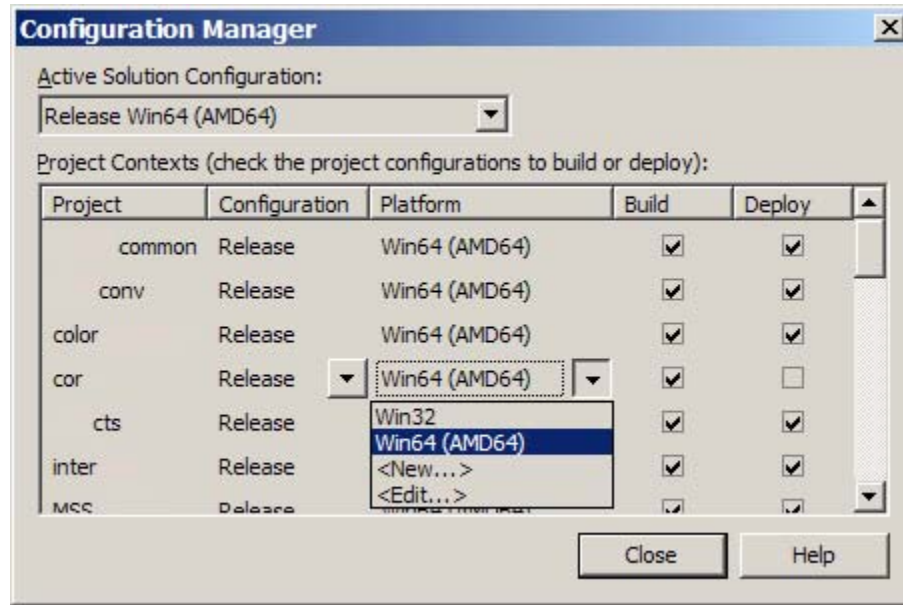


Figure 6: Microsoft Visual Studio “Whidbey” projects configuration.

Then select each of the newly created configurations in “Active Solution Configuration” – recognizable by the suffix “Win64 (AMD64)” – and select the platform “Win64 (AMD64)” for each sub-project (see Figure 6).

Microsoft Visual Studio “Whidbey” is still just a preview of what the next-generation development environment will be, so there are still some steps that need to be performed that probably will not be necessary when it is officially released. Navigate each sub-project that make up the solution and edit select “Properties” in the menu brought up by a right-click:

- In all configurations for the AMD64 platform, in the “General” branch, make sure to change the output directories in order to avoid confusing the environment by potentially mixing AMD64 and Win32 object files.
- In all configurations for the AMD64 platform, in the “C/C++ | General” branch, make sure that if the “Debug Information Format” is enabled, it is not “Program Database for Edit and Continue”.
- In the debug configuration for the AMD64 platform, in the “C/C++ | General” branch, it is a good idea to add the option `/Wp64` to the options list to help to make the source code 64-bit compliant.

Now, select the menu options “Build | Rebuild Solution” to start porting the application to the AMD64 architecture.

Debugging AMD64 Applications

Unlike the previous versions, Microsoft Visual Studio “Whidbey” features full support to debug applications on an AMD64 system running 64-bit Windows for AMD64 processors.

Select a sub-project for a program and choose “Properties” in the menu that pops up when it is right-clicked. In the “Debugging” branch:

- Set the “Connection” type to “Remote via TCP/IP”.
- Set the “Remote Machine” name to its hostname.

- Set “Remote Command” to the same contents as in “Command”, however the path needs to be changed to something suitable to the remote system.

Next, copy the Microsoft Remote Debug Monitor to the remote system, which can be normally found at “C:\Program Files\Microsoft Visual Studio .NET Whidbey\Common7\IDE\Remote Debug Components (AMD64)” (all files should be copied over). The files “msvcr80.dll” and “msvcp80.dll” from this directory should be copied to “%systemroot%\system32” on the remote system.



Figure 7: Microsoft Remote Debug Monitor.

Finally, start the Microsoft Remote Debug Monitor on the remote system with the following options (see Figure 7):

```
msvsmon.exe -noauth -anyuser
```

At this point, it is now possible to use the built-in debugging features of Microsoft Visual Studio “Whidbey” on projects for the AMD64 platform, from starting and pausing execution to setting breakpoints and watchpoints (see Figure 8).

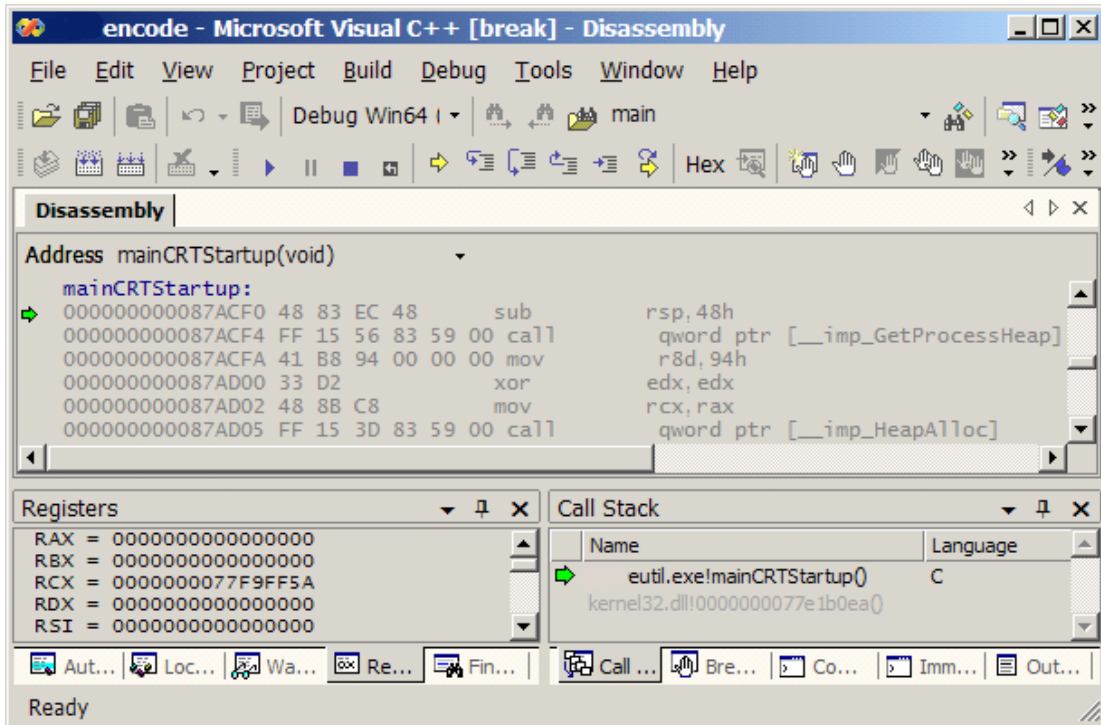


Figure 8: Microsoft Visual Studio “Whidbey” remote debug session on the AMD64 platform.

Call to Action

Microsoft provides the developers with all the tools necessary to migrate to 64-bit Microsoft Windows for AMD64 processors. It is now up to the developers to port their applications and drivers to the AMD64 architecture and enjoy the benefits of 64-bit computing.

AMD provides rich information in the form of guides and white-papers on real-world examples on the AMD64 architecture (see Resources and References) as well as the free profiling tool for AMD processors, CodeAnalyst, which fully supports the AMD64 architecture (see http://www.amd.com/us-en/Processors/DevelopWithAMD/0,,30_2252_3604,00.html).

Moreover, through the AMD Development Center (<http://www.developwithamd.com/>), developers can have access to remote AMD Opteron™ processor-based systems to test their applications while counting on technical support directly from AMD.

Resources and References

- Tools and resources from Microsoft:
 - Microsoft Platform SDK:
<http://www.microsoft.com/msdownload/platformsdk/sdkupdate>
 - Microsoft Windbg: <http://www.microsoft.com/whdc/ddk/debugging>
 - Microsoft Windows Driver Development Kit:
<http://www.microsoft.com/whdc/ddk/winddk.mspix>
 - Microsoft Windows XP for AMD64 processors:
<http://www.microsoft.com/windowsxp/64bit/downloads/upgrade.asp>

- Tools and resources from AMD:
 - Software Optimization Guide for AMD64: http://www.amd.com/us-en/assets/content_type/DownloadableAssets/dwamd_25112.pdf
 - AMD Bios and Kernel Developer's Guide: http://www.amd.com/us-en/assets/content_type/DownloadableAssets/dwamd_26049.pdf
 - AMD64 Porting FAQ: http://www.amd.com/us-en/assets/content_type/DownloadableAssets/dwamd_AMD64_Porting_FAQ.pdf
 - AMD64 Architecture Programmer's Manuals: http://www.amd.com/us-en/Processors/DevelopWithAMD/0,,30_2252_875_4622,00.html
 - AMD CodeAnalyst: http://www.amd.com/us-en/Processors/DevelopWithAMD/0,,30_2252_3604,00.html
 - AMD; "AMD64 Porting FAQ"; http://www.amd.com/us-en/assets/content_type/DownloadableAssets/dwamd_AMD64_Porting_FAQ.pdf.
 - AMD; "AMD64 Programming Documentation"; http://www.amd.com/us-en/Processors/DevelopWithAMD/0,,30_2252_9044,00.html#amddocs.
 - AMD64 DevSource: <http://www.devx.com/amd/>

- Related white-papers:
 - Jagasia, Harsha; AMD; "Optimizing Codecs for Microsoft Windows for the AMD64 Architecture"; <http://www.microsoft.com/whdc/winhec/> (WinHEC 2004 Papers).
 - Wall, Mike; AMD; "Porting and Optimizing Applications on 64-bit Microsoft Windows for the AMD64 Architecture"; <http://www.microsoft.com/whdc/winhec/> (WinHEC 2004 Papers).

About the Author and Acknowledgments

Evandro Menezes works in the Software Research and Development's Tools Team at AMD in Austin, Texas. Evandro has been working with the optimization and porting of tools, libraries and applications for the AMD64 architecture since the AMD Opteron[™] processors were still in the development stage.

Evandro wants to thank all those who supported him in the writing of this paper: Jim Conyngham, Chip Freitag Frank Gorishek, Mark Santaniello and Mike Wall; in particular, Tom Deneau for suggesting capturing the author's experience working with Microsoft tools on AMD64 processors and Harsha Jagasia for thoroughly reviewing this paper.