



Building AMD64 Applications with the Microsoft[®] Platform SDK

Developer Application Note



Publication # 30887	Revision: 3.00
Issue Date: October 2003	

© 2003 Advanced Micro Devices, Inc. All rights reserved.

The contents of this document are provided in connection with Advanced Micro Devices, Inc. (“AMD”) products. 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 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. Except as set forth in AMD’s Standard Terms and Conditions of Sale, 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.

AMD’s products are not designed, intended, authorized or warranted for use as components in systems intended for surgical implant into the body, or in other applications intended to support or sustain life, or in any other application in which the failure of AMD’s product could create a situation where personal injury, death, or severe property or environmental damage may occur. AMD reserves the right to discontinue or make changes to its products at any time without notice.

Trademarks

AMD, the AMD Arrow logo, AMD Athlon, AMD Opteron, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Microsoft and Windows are registered trademarks of Microsoft Corporation.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Contents

Revision History	3
Chapter 1 Using the Microsoft® Platform SDK for AMD64 Development	5
1.1 Audience	5
1.2 Technical Preliminaries	5
1.3 In This Book	6
Chapter 2 Using Visual Studio .NET on Win32 to Develop 64-Bit Applications	7
2.1 Configuring Your 32-Bit IDE	7
Chapter 3 Developing 64-Bit Windows® Applications with Visual Studio 6	9
3.1 Installing the Windows Platform SDK	9
3.1.1 Win32-Based Systems	9
3.1.2 Installing the Windows Platform SDK on AMD64 Processor-Based Systems	9
3.2 Setting up Visual Studio 6 IDE to Compile 64-Bit Code for AMD64 Processor-Based Systems	10
3.3 Build an AMD64 Application with the Visual Studio 6 IDE	10
3.4 Start Visual Studio 6 with 64-bit build environment.	11

Revision History

Date	Revision	Description
October 2003	3.00	Initial release.

Chapter 1 Using the Microsoft® Platform SDK for AMD64 Development

Microsoft Visual Studio 6 with Service Pack 4 or later (including Visual Studio 6 with Service Pack 5, Visual Studio .NET, and Visual Studio .NET 2003) may be used to develop 64-bit applications for AMD64 platforms using the Microsoft® Platform Software Development Kit (SDK). These development environments may also be used to port existing 32-bit Visual Studio projects to a 64-bit operating environment. The principal complications arising when using these IDEs involve the proper installation of the Platform SDK and configuration of the working environment and development software for compiling and building 64-bit AMD64 applications. This document addresses these interrelated issues.

1.1 Audience

This document is intended for the programmer who is creating 64-bit software for deployment in a 64-bit Microsoft Windows® operating environment on AMD Athlon™ 64 or AMD Opteron™ processor-based platforms.

1.2 Technical Preliminaries

The Windows Platform SDK supports platform development for AMD64 systems running the Microsoft Windows 64-bit operating system for AMD64 processor-based systems. The Microsoft Platform SDK includes the 64-bit compiler/linker and other tools, along with the C run-time (CRT), Microsoft Foundation Class (MFC) and Active Template (ATL) libraries. You can obtain the Platform SDK for AMD64 platforms free of charge from Microsoft on MSDN or Betaplace, as part of the AMD64 64-bit Windows beta program.

Once you have the Platform SDK, there are two approaches from which to choose:

- You can install the Platform SDK on any PC running 32-bit Windows, and cross-compile your source code as an AMD64 application for deployment and testing on a separate AMD64 processor-based 64-bit Windows “target” installation. AMD recommends this development environment, although it requires two separate PCs or two boot partitions on an AMD64 processor-based machine: a 32-bit development system and a 64-bit test platform.
- You can install the Platform SDK directly on an AMD64 processor-based 64-bit machine, so you can build and run the application on the same machine.

The components of the Platform SDK assume that the PATH, INCLUDE, and LIB environment variables have certain values that are different from those assumed by the tools provided in Visual Studio and Visual Studio .NET. To build a 64-bit application, the Platform SDK environment variable settings must be passed to Visual Studio 6 or Visual Studio .NET for use during the build process. Visual Studio 6 (SP4 and SP5), Visual Studio .NET, and Visual Studio .NET 2003 allow the

development environment to be started from the command line. Command line options (switches) allow the user to open Visual Studio or Visual Studio .NET with special configuration information. One such switch is `/USEENV`. (Command-line switches are not case sensitive.)

Starting Visual Studio 6 (Service Pack 4 or 5) or Visual Studio .NET on the command line with the `/USEENV` switch superimposes the compiler, linker, and path settings from the environment over the usual system-wide IDE default settings. This enables Visual Studio 6 or Visual Studio .NET to use the Platform SDK's predefined environment settings, which are accessible from the START menu and allows the use of the 64-bit tools, libraries and header files from the Platform SDK to build applications for 64-bit Windows.

1.3 In This Book

Chapter 2 describes the procedures used to build your application with the Platform SDK using Microsoft Visual Studio .NET on a Windows 32-bit development system (Win32 host).

Chapter 3 describes the use of the Platform SDK to develop or port 64-bit applications in a Windows environment running on either a 32-bit or 64-bit AMD64 processor-based system using Microsoft Visual Studio 6.

Chapter 2 Using Visual Studio .NET on Win32 to Develop 64-Bit Applications

This chapter describes the use of the Platform SDK to develop 64-bit applications in a 32-bit Windows® environment using Microsoft® Visual Studio .NET. After compilation and building, 64-bit applications can be installed on systems running a 64-bit Windows operating system on AMD Athlon™ 64 or AMD Opteron™ processor-based systems. This document describes the preliminary steps necessary to convert an existing 32-bit Visual Studio .NET application to 64 bits.

2.1 Configuring Your 32-Bit IDE

First, install Visual Studio .NET on your 32-bit Windows development computer. Also install the Windows Platform SDK that supports AMD64 development. Next steps:

Step 1. Open the command console to set up the pre-defined environment for a 64-bit Retail build.

1. On the **Start** menu, find the **Microsoft Platform SDK** group.
2. Choose **Open Build Environment Window** submenu.
3. Choose the **Windows Server 2003 64-bit Build Environment** submenu.
4. Select **Set Win Svr 2003 AMD Build Env (Retail, PreRelease)**.
5. This command opens a console window with PATH, INCLUDE, and LIB build environment variables set for a 64-bit build.

Step 2. Start Visual Studio .NET at the command line using command line options.

1. Change directories at the console command prompt:
cd "C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\IDE"
2. Start Visual Studio .NET from the command line with the /USEENV command switch:
devenv /useenv
Visual Studio .NET opens with PATH, INCLUDE, and LIB environment variables set for 64-bit compilation and linking.

Step 3. Open an existing project and add a configuration "Release64."

Task 1—Navigate to your project subdirectory.

1. Open your 32-bit Visual Studio project.

Task 2—Add a new 64-bit configuration to the existing project, if necessary.

1. In the **Build** menu choose **Configuration Manager**.
2. Select the drop down for **Active Solution Configuration**.
3. Select **New...** to create a new configuration.
4. Add a new configuration based on **Release** settings and call it **Release64**.
5. Notice that the default configuration is now the 64-bit choice.
6. Close the configuration dialog.

Step 3. Change the Release64 configuration options for 64-bit build.

To compile the 64-bit application, you must change the various compiler and linker options that are specific to a given build configuration.

Task 1—Change the build options to compile a 64-bit application.

1. Highlight your project file in the Visual Studio **.NET Solution Explorer** window.
2. Click on the **Property Pages** icon in the tool bar of the **Properties** window.
This displays the **Property Pages** window for your project.
3. Go to the C/C++ node and highlight **General**.
4. Under **General** settings change the **Debug Info** format to **Program Database (/ZI)**.
5. Go to the **Linker** node.
6. Go to the **Command Line** settings section.
This displays the currently available command-line options and the **Additional Options** text box.
7. Type **/machine:AMD64** in the **Additional Options** text box and click **OK**.
This tells the linker which libraries to choose.

You can now build your 64-bit Release code.

Task 2—Build 64-bit code with 32-bit Visual Studio .NET

1. From the **Build** menu select **Build Solution**.
2. The code should compile and link successfully.
3. You can now copy the executable to the 64-bit target system and run it.

Chapter 3 Developing 64-Bit Windows® Applications with Visual Studio 6

Some Visual Studio IDEs are subject to certain restrictions when running on 64-bit AMD64 platforms. In particular, Microsoft® Visual Studio .NET is not currently available for use on AMD64 processor-based systems. The following procedures describe configuration considerations for developing 64-bit applications using Microsoft Visual Studio 6 (Service Pack 4 or 5) on 64-bit AMD64 platforms under a Microsoft 64-bit Windows® operating system. You may also follow the same procedures to set up a development system on 32-bit Windows running Visual Studio 6 to develop 64-bit applications for testing on a 64-bit AMD64 processor-based system.

3.1 Installing the Windows® Platform SDK

The process of installing the Windows Platform SDK for AMD64 on a 64-bit Windows host is different from the installation procedure on a 32-bit host. The following procedures assume a CD-ROM installation.

3.1.1 Win32-Based Systems

First, install Visual Studio 6 (with Service Pack 4 or 5) on your 32-bit Windows development computer. Then install the Windows Platform SDK that supports AMD64 development. This Platform SDK is available for download on Microsoft MSDN and Betaplace or on CD-ROM.

3.1.2 Installing the Windows® Platform SDK on AMD64 Processor-Based Systems

The top-level installer will probably not run under Win64; this is expected. To work around this:

1. Right-click the CD-ROM icon and select **Explore**.
2. Navigate into the `\setup` directory, and install the following three Microsoft Windows Installer components by double-clicking them and accepting all the default settings:

CoreSDK-x86.msi

MDACSDK-x86.msi

PSDK-x86.msi

This installs the AMD64 tools and libraries.

Note: You may try installing additional components by double-clicking their **.msi** files, if some files required by your application appear to be missing.

If you are comfortable using command-line tools, you can now use the Platform SDK tools from the command line. If you have an existing Visual Studio 6 project, load it into Visual Studio 6 and use **Project/Export Makefile...** to write a makefile, then use the command-line tools with that makefile

as input. You can still keep your Visual Studio 6 project as the master, editing the project from the IDE, and generating new makefiles when necessary. If you choose to use the command-line tools this way, you can skip the remainder of this document.

3.2 Setting up Visual Studio 6 IDE to Compile 64-Bit Code for AMD64 Processor-Based Systems

Once you have the AMD64 Platform SDK installed, you can use the Visual Studio 6 IDE to build 64-bit applications. This is convenient if you have existing 32-bit applications that you would like to migrate to 64-bit, because this allows you to maintain a 64-bit build as a different build configuration of the same project. This leaves you with only one set of source files to manage.

After you have installed these products, take the following steps to assure that Visual Studio 6 uses the AMD64 compiler.

3.3 Build an AMD64 Application with the Visual Studio 6 IDE

Because of basic differences in the installation of the Platform SDK, the build process is different when using Visual Studio 6 on a WIN32 system than when using an AMD64 processor-based system. The first task is to open a DOS command window.

Task 1—Set the build options to compile a 64-bit application.

Development on 32-Bit Windows:

1. On the **Start** menu, find the **Microsoft Platform SDK** group.
2. Choose **Open Build Environment Window** submenu.
3. Choose the **Windows Server 2003 64-bit Build Environment** submenu.
4. Select **Set Win Svr 2003 AMD Build Env (Retail, PreRelease)**.
5. This command opens a console window with PATH, INCLUDE, and LIB build environment variables set for a 64-bit build.

Note: Do not close this command window!

Development on 64-bit Windows:

1. Set the 64-bit build environment.
 - a. On the **Start** menu, find **Run** and enter “**cmd**”.
This opens a command console window.
 - b. Change to the **\Program Files (x86)\Microsoft SDK** directory.
 - c. Run the **SetEnv.bat** batch program.

SetEnv /AMD64 /DEBUG

or

SetEnv /AMD64 /RETAIL

Ignore any warning message such as “\Microsoft was unexpected at this time.”

The **SetEnv.bat** batch file sets the appropriate environment variables for the Platform SDK build environment with respect to operating system and platform type.

Note: Do not close this command window!

3.4 Start Visual Studio 6 with 64-bit build environment.

1. In the same command window as used in Step 1, start Visual Studio 6 by running the following command:

msdev /useenv

This command starts the Visual Studio 6 IDE with the include, library and executable directories set for the 64-bit build environment. If **msdev.exe** is not in the path, navigate to the **..\Microsoft Visual Studio\Common\msdev98\bin** directory before running **msdev.exe**.

2. In Visual Studio, open your project.
3. Add a 64-bit Debug or Retail configuration, if necessary.
 - a. On the **Build** menu, choose **Configurations**.
 - b. In the **Configurations** dialog, click the **Add** button.
 - c. In the **Add Project Configuration** dialog, set the **Configuration** to **Debug64 | Retail64** and set **Copy Settings from** to *Your Project Name – Win32 Debug | Retail*.
 - d. Click **OK** button and then the **Close** button to add the configuration.
4. Set the active configuration to 64-bit.

On the **Build** menu, choose **Set Active Configuration** to *Your Project Name - Win32 Debug64 | Retail64*.

5. Modify project settings.

There are a few Visual Studio 6 compiler/linker options that do not apply to a 64-bit compile or link. They must be modified to avoid errors or warnings.

- a. In the **Project Settings**, select the **General** tab and change the Intermediate files and Output files directories to **Debug64 | Retail64**.
- b. In the **Project Settings**, select the **C/C++** tab. Set the **Debug Info** to **Program Database** (compiler option, **/ZI**).
- c. In the **Project Settings**, select the **C/C++** tab. In the **Project Options**, remove the **/GZ** option. This disables runtime checking.

- d. In the **Project Settings**, select the **Link** tab. In the **Project Options**, change “/machine:I386” to “/machine:AMD64”. (You may need to scroll down to see this.)
 - e. In the **Project Settings**, click **OK**
 - f. In the **Workspace** window (left edge of screen) select the **File View** tab, highlight the *Your Project Name.hpj* file and press the **Del** key to delete this file from the project, if it exists.
 - g. From the **Tools** menu select **Options**. Under the **Directories** tab, select **Library Files** from the **Show directories for:** drop-down box. Add the paths: **C:\Microsoft SDK\lib\AMD64** and **C:\Microsoft SDK\lib\AMD64\mfc**, if not listed. Delete all other library paths.
6. Build the project.

Now you have a 64-bit application that is ready to be run on an AMD64 system with Win64. Note that you can run an **.exe** file from the Visual Studio 6 IDE. Follow steps below to do so. Note that you cannot debug the **.exe** from the Visual Studio 6 IDE. To debug your project refer to step 8.
 7. To run your application from inside the Visual Studio IDE, run the program using the **Build/Execute** menu.
 8. To debug, use **WINDBG** or **NTSD** which are included with Win64 or the Microsoft Windows Driver Development Kit (DDK), both available from MSDN.