

Getting Started with the Intel® Parallel Inspector

This guide shows you how to use basic Intel® Parallel Inspector features to identify and analyze a threading and memory issue. It uses simple code samples to show how to:

- Set up and run a threading or memory error analysis.
- Locate threading or memory errors.
- Manage the analysis and interpret results.
- Resolve the issues.

NOTE: For a video demonstration on getting started with the Intel® Parallel Inspector, try the [Show Me videos](#). Show Me videos require Adobe* Flash* Player.

Contents

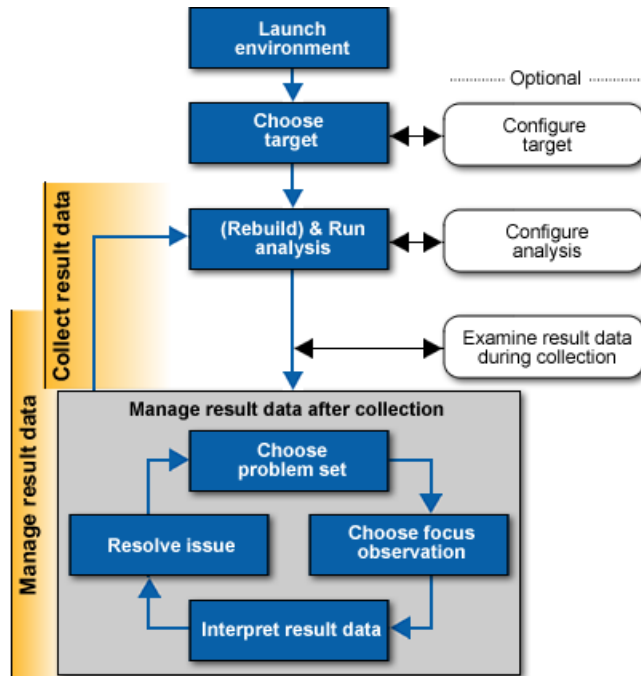
1	Overview	2
2	Choose and Build Target	3
3	Threading Errors-Collect and Manage Result	4
4	Memory Errors-Collect and Manage Result	9
5	User and Reference Documentation	13
	Legal Information	14

1 Overview

Basic Workflow

This guide demonstrates how to identify and analyze a threading error and memory error within the framework of a basic workflow.

This workflow is also the organizing principle for Intel® Parallel Inspector Help.



Basic Intel® Parallel Inspector Workflow

Guide Objective

Help you identify and analyze a threading and memory error that prevent a sample application from correctly displaying an `abcde` banner.

```

    aaa      b      ccc      d      eee
    a a      bbb     c       ddd     eee
    a a a    b b     c       d d     e
    aaaaa   bbb     ccc     ddd     eee
  
```

Sample Banner Application

Tools

Use the following tools to achieve the guide objective:



- Visual Studio* 2005 or 2008 development environment
- The Intel® Parallel Inspector
- Intel® Parallel Inspector Help

To display Help, from the Visual Studio* Help menu, choose **Intel Parallel Studio > Parallel Studio Help > Inspector Help**. While using Intel® Parallel Inspector in the Visual Studio* environment, you can also interactively display relevant Help by pressing the F1 key, clicking a ? button, or clicking a link in the **Dynamic Help** window.

- Sample applications shipped with the Intel® Parallel Inspector

NOTE: All code examples are designed only to illustrate Intel® Parallel Inspector features. They do not represent best practices for creating multithreaded code. Results may vary depending on the nature of the analysis.

Preparation

1. Copy the samples .zip file from the `samples\<locale>` folder in the Intel® Parallel Inspector installation folder (the default installation folder is `C:\Program Files\Intel\Parallel Studio\Inspector`) to a writable directory or share on your system, such as a `My Documents\Visual Studio 200x\Intel\samples` folder.
2. Extract the sample files from the .zip file.

2 Choose and Build Target

NOTE: This guide presents screen shots from the Visual Studio* 2005 development environment.

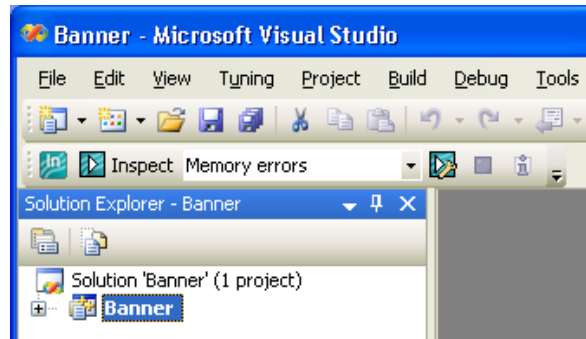
NOTE: The sample applications are non-deterministic. Your screens may vary from the screen shots shown throughout this guide.



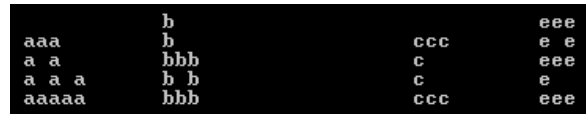
A *target* is an executable file you inspect for threading and memory errors using the Intel® Parallel Inspector.

See **Choosing Targets > About Choosing Targets** in Inspector Help for information about compiler/linker options that produce the most accurate result data.

1. From the Microsoft Visual Studio* 2005 or 2008 menu, choose **File > Open > Project/Solution**.
2. In the **Open Project** dialog box, open the banner\Banner.sln file to display the solution/project in the **Solution Explorer**.
3. Ensure the **Banner** project is highlighted.
4. From the Visual Studio* menu, choose **Debug > Start Without Debugging** to compile, link, and test the project.
5. Check **Output** to ensure the project compiled successfully.
6. Check for a console window with a banner display. This ensures the project runs on your system without the Intel® Parallel Inspector. Press the Enter key twice to close the console window.



Banner - 0 error(s), 0 warning(s)
***** Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped *****

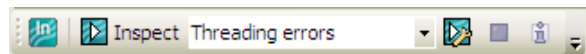


3 Threading Errors-Collect and Manage Result

Configure and Run Analysis

1. From the Visual Studio* menu, choose **Tools > Intel Parallel Inspector > Inspect Threading Errors** to display the **Configure Analysis** dialog box.

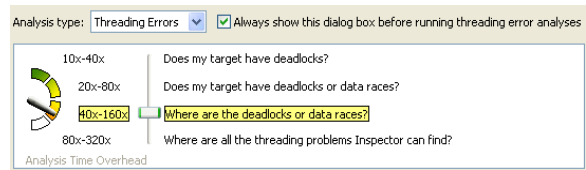
NOTE: This menu command is also available from the Inspector toolbar:



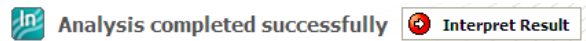
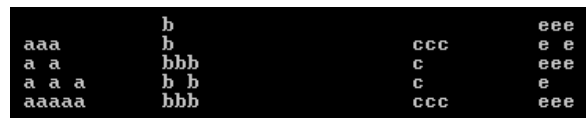


See **Configuring Analyses > About Configuring Analyses** in Inspector Help for information about preset configuration levels.

2. Ensure **Threading Errors** appears in the **Analysis type** drop-down list.
3. Drag the configuration slider to the **Where are the deadlocks or data races?** preset level.
4. Click the **Run Analysis** button to execute the target and collect a result in a `My Inspector Results\resultdir*.insp` file in the solution folder.
5. Notice the banner displays incorrectly (`ab ce` instead of `abcde`).
6. Press the Enter key to end target execution and close the console window.
7. On the **Event Log** window, click the **Interpret Result** button to start managing result data.



Example: `My Inspector Results\r000-ti3\r000-ti3.insp`, where `ti` = threading error analysis type and `3` = third preset level.



Choose Problem Set

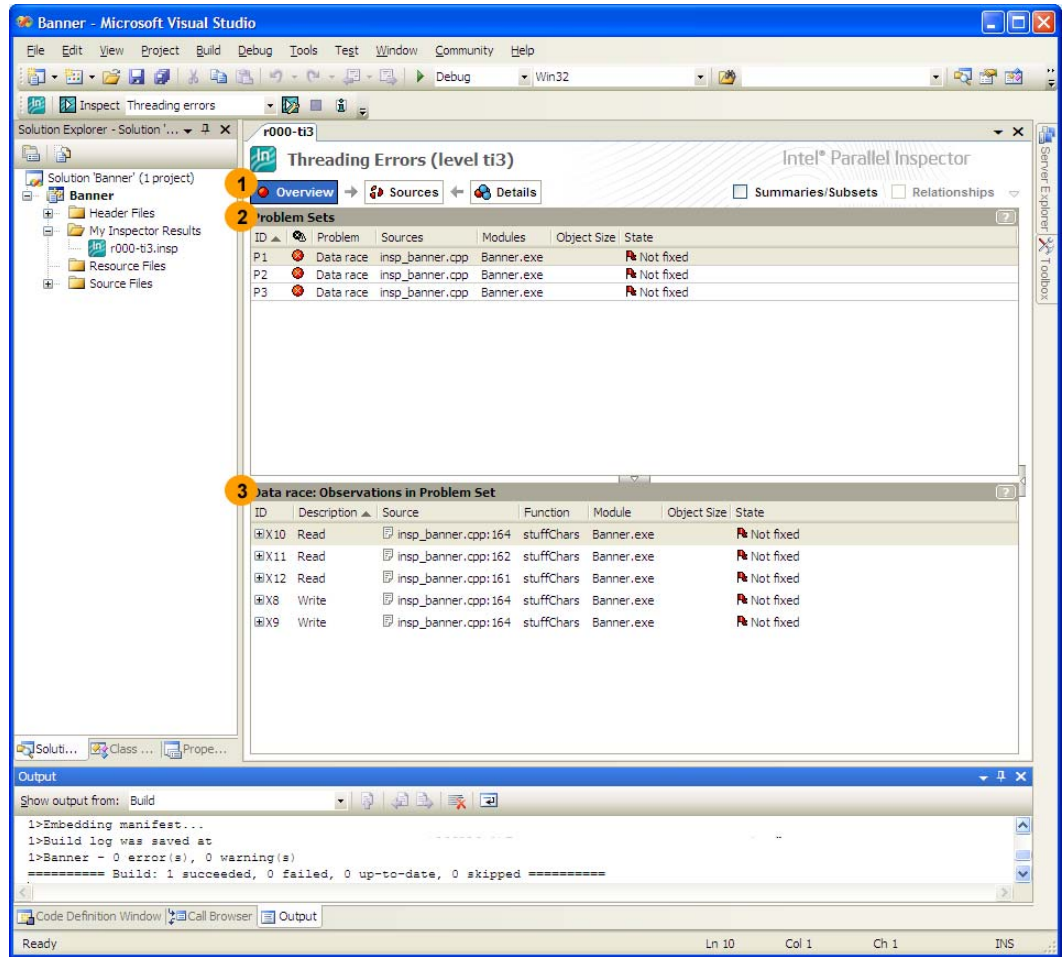
An *observation* is a fact about target source code.

A *problem* is a small group of closely related observations that indicate an error.

A *problem set* is a larger group of more loosely related observations that could share a common solution.

Data race problems occur when:

- Multiple threads write to the same memory location without proper synchronization.
- Or one thread reads while a different thread concurrently writes to the same memory location without proper synchronization.
- Or one thread writes to while a different thread concurrently reads from the same memory location.



- 1 The **Overview** window is the default starting point for managing result data.
- 2 The **Problem Sets** pane prioritizes problem sets by severity and size. Think of the **Problems Sets** pane as a *to-do* list. Start at the top and work your way down.
- 3 The Intel® Parallel Inspector selects the first problem set for you by default, and displays all the observations in all the problems in this problem set in the **Observations in Problem Set**.

Choose Focus Observation

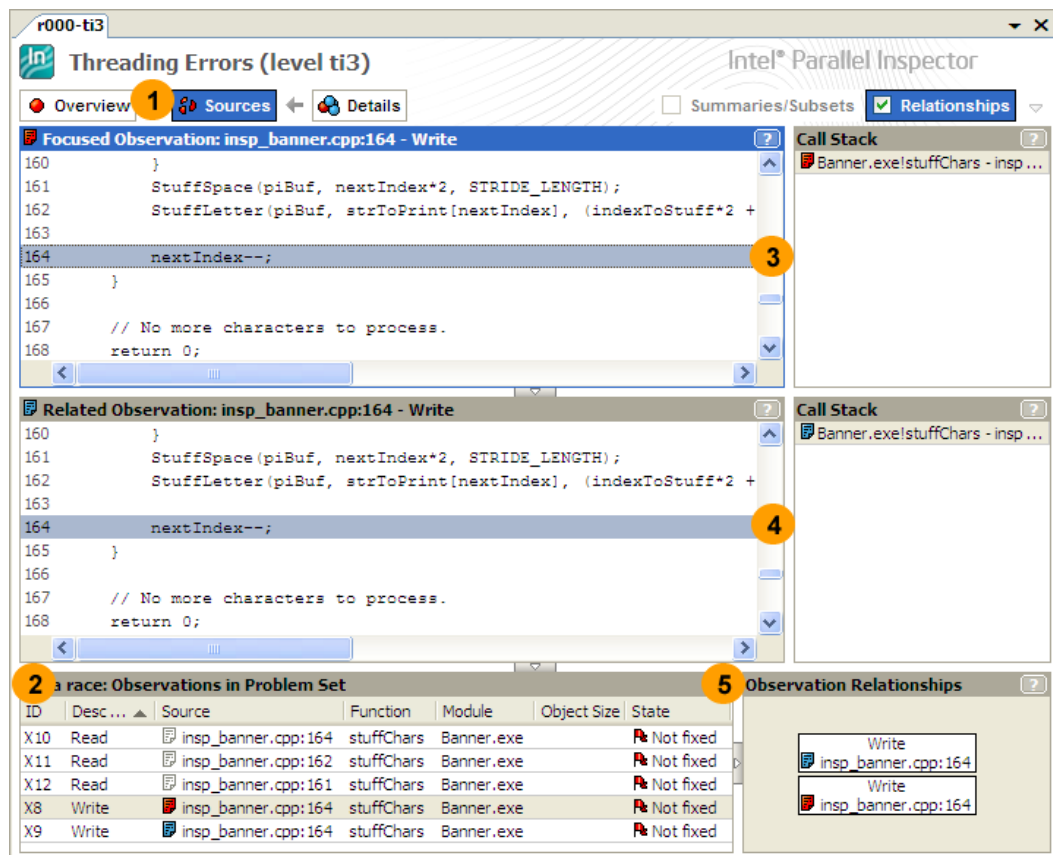
A *focus observation* is an observation with relationships you choose to explore.

Perhaps the problem type description—in this case, **Data race**—and the observation type descriptions—in this case, **Read** and **Write**—are all you need to know to resolve



an issue. Maybe viewing the source code for one or more observations is all you need: click a **+** icon to display a source code snippet. But sometimes you need to understand the relationships among observations to resolve an issue. Double-click the first **Write** observation in the **Observations in Problem Set** pane to display the **Sources** window and explore its relationship to other observations in the problem set.

Interpret Result Data



- 1 The **Sources** window offers a variety of devices to explore relationships among observations.
- 2 The **Observations in Problem Sets** pane displays all the observations in all the problems in the problem set.
- 3 The **Focus Observation Code** and **Focus Observation Call Stack** panes show the source code and parent location(s) for the selected focus observation.

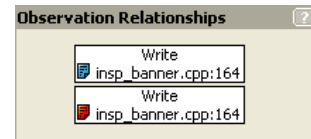
- 4 The **Related Observation Code** and **Related Observation Call Stack** panes show the source code and parent location(s) for one observation related to the focus observation. To display source code for a different related observation, click the desired observation in the **Observations in Problem Set** pane.
- 5 The **Observation in Relationships** pane graphically shows the relationship between the current focus and related observations in a problem in the problem set.

Notice the and icons throughout the **Sources** window.

Icon	Meaning
	This observation is the focus observation. It has source code available for viewing in the Intel® Parallel Inspector and editing in the Visual Studio* code editor. That source code is currently displayed in the Focus Observation Code pane (at the top of the Sources window).
	This observation is related to the focus observation. It has source code available for viewing and editing, and that source code is currently displayed in the Related Observation Code pane (in the middle of the Sources window).
	This observation has source code available for viewing and editing.
None	This observation does not have source code available for viewing and editing.

Right-click the diagram to display a pop-up menu, then choose **Explain Problem** to see *Inspector Help* information for this problem type.

Look at the diagram in **Observation Relationships** pane.



In relationship diagrams:

- Each box in a diagram represents an observation in a problem.
- A diagram with a single box is a trivial problem with no related observation.
- Vertically stacked boxes indicate concurrent observations.
- Boxes arranged left-to-right with connecting arrows indicate a time ordering.
- Boxes with connecting lines indicate association.

This diagram clearly identifies the issue: a data race where two threads write to the same memory location without proper synchronization.

Resolve Issue

Under standard debugging circumstances, you would now do the following:

1. Review the source code in the **Focus Observation Code** and **Related Observation Code** panes.
2. Double-click the source code you want to edit.



Double-clicking opens the source file in a new tab where you can edit it with the Visual Studio* code editor.

3. When you're finished editing source code for this problem, return to the result tab and click another observation in the **Observations in Problem Set** pane to explore other problems in the problem set. Edit source code as necessary.
4. When you're finished editing source code, click the **Overview** button on the result tab to return to the **Overview** window.
5. Manage the remaining problem sets using the same techniques you used to manage the first problem set.
6. When you're finished managing problem sets, rebuild the target and rerun the analysis: from the Visual Studio* menu, choose **Tools > Intel Parallel Inspector > Re-inspect**.

NOTE: The preset configuration level is persistent by analysis type by target until you reconfigure.

If you handled all threading issues appropriately, the Intel® Parallel Inspector displays no problems sets; however, the banner still displays incorrectly because the **Banner** target still contains errors—memory errors.

Instead, proceed to the *Memory Errors-Collect and Manage Result* section.

4 *Memory Errors-Collect and Manage Result*

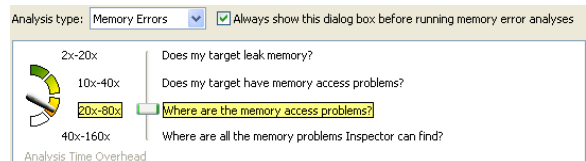
The process for collecting and managing a memory result is similar to that for collecting and managing a threading result. The key difference is your selection in the **Configure Analysis** dialog box.

NOTE: You can also use the Intel® Parallel Inspector to check for memory errors in single-threaded applications.

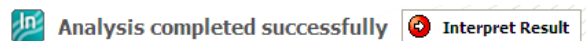
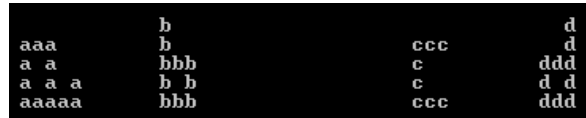


Configure and Run Analysis

1. From the Visual Studio* menu, choose **Tools > Intel Parallel Inspector > Inspect Memory Errors** to display the **Configure Analysis** dialog box.
2. Ensure **Memory Errors** appears in the **Analysis type** drop-down list.
3. Drag the configuration slider to the **Where are the memory access problems?** preset level.
4. Click the **Run Analysis** button to execute the target and collect a result in another `My Inspector Results\resultdir*.insp` file in the solution folder.
5. Notice the banner displays incorrectly (`ab cd` instead of `abcde`).
6. Press the Enter key to end target execution and close the console window.
7. On the **Event Log** window, click the **Interpret Result** button to start managing result data.

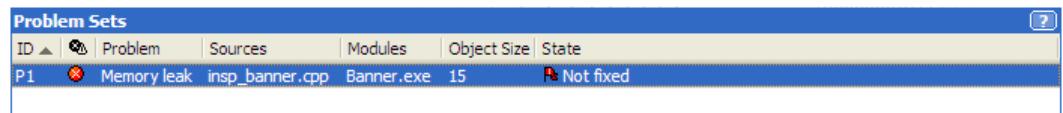


Example: `My Inspector Results\r000-mi3\r000-mi3.insp`, where `mi` = memory error analysis type and `3` = third preset level.



Choose Problem Set

Memory leak problems occur when a block of memory is allocated and never released.



The Intel® Parallel Inspector selects the single problem set for you by default.



Choose Focus Observation

The **Allocation site** observation represents the location and associated call stack from which the memory block was allocated.

Memory leak: Observations in Problem Set						
ID	Description	Source	Function	Module	Object Size	State
X1	Allocation site	insp_banner.cpp:52	main	Banner.exe	15	Not fixed

Double-click the **X1** observation.

Interpret Result Data

The screenshot shows the Intel Parallel Inspector interface. The main window displays the source code for `insp_banner.cpp` with line 52 highlighted: `char* prefix = (char*)malloc(15);`. The `Memory Errors (level mi3)` pane at the bottom shows the `Memory leak: Observations in Problem Set` table with the `X1` observation selected. The `Observation Relationships` pane shows the selected observation's relationship to the allocation site.

Memory leak: Observations in Problem Set						
ID	Description	Source	Function	Module	Object Size	State
X1	Allocation site	insp_banner.cpp:52	main	Banner.exe	15	Not fixed



Resolve Issue

Under standard debugging circumstances, you would now do the following:

1. Double-click the **Focus Observation Code** pane to open the source file in a new tab where you can edit it with the Visual Studio* code editor.
2. When you're finished editing source code, click the **Overview** button on the result tab to return to the **Overview** window.
3. Rebuild the target and rerun the analysis: from the Visual Studio* menu, choose **Tools > Intel Parallel Inspector > Re-inspect**.

If you handled the memory issue appropriately, the Intel® Parallel Inspector displays no problems sets.

Instead, examine an Intel® Parallel Inspector result when there are no threading and memory errors.

1. Close the **Banner** solution/project.
2. Open the `banner\BannerFixed.sln` file.
3. Build the **BannerFixed** target.
4. Configure and run a threading analysis.
 - Notice the banner displays properly.
 - When you press any key to end target execution and close the console window, and click the **Interpret Result** button on the **Event Log** window, the **Overview** window shows no problem sets.
5. Configure and run a memory analysis.
 - Notice the banner displays properly.
 - When you press any key to end target execution and close the console window, and click the **Interpret Result** button on the **Event Log** window, the **Overview** window shows no problem sets.
6. If desired, use a diffing tool of your choice to compare the `banner\insp_banner.cpp` and `banner\insp_bannerfixed.cpp` files.

NOTE: As a next step, read the *Intel® Parallel Inspector Sample Code Guide*. It uses additional code samples to show you how to resolve five common threading and memory issues.



5 User and Reference Documentation

This guide focuses on basic features of the Intel® Parallel Inspector. To explore more features and get the most out of the Intel® Parallel Inspector, check the following resources:

Resource	Notes
<i>Intel® Parallel Inspector Documentation</i>	Use this HTML page to locate other Intel® Parallel Inspector resources. To open this HTML page, from the Windows* Start menu, choose Intel Parallel Studio > Parallel Studio Documentation > Inspector Documentation .
Sample code	Use sample code in a zip file provided at <code><install_dir>\samples</code> to learn how to resolve common threading and memory errors. Read the associated Sample Code Guide available at <code><install_dir>\documentation\<locale></code> .
Intel® Parallel Studio resources	<p>Intel Parallel Studio provides the most comprehensive set of tools for parallelism including Intel® Parallel Amplifier, Intel® Parallel Composer, and Intel® Parallel Inspector.</p> <p>Refer to the <i>Getting Started with the Parallel Studio</i> for an overview of all the components in the Parallel Studio. From the Windows* Start menu, choose Intel Parallel Studio > Getting Started > Parallel Studio Getting Started Guide.</p> <p>To open documentation that points to more resources for each installed Intel Parallel Studio tool, from the Windows* Start menu, choose Intel Parallel Studio > Parallel Studio Documentation > Inspector Documentation.</p> <p>You can find additional information on the Intel® Parallel Studio at http://www.intel.com/software/products</p>



Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel's Web Site.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See http://www.intel.com/products/processor_number for details.

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino Atom, Centrino Atom Inside, Centrino Inside, Centrino logo, Core Inside, FlashFile, i960, InstantIP, Intel, Intel logo, Intel386, Intel486, IntelDX2, IntelDX4, IntelSX2, Intel Atom, Intel Atom Inside, Intel Core, Intel Inside, Intel Inside logo, Intel. Leap ahead., Intel. Leap ahead. logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Viiv, Intel vPro, Intel XScale, Itanium, Itanium Inside, MCS, MMX, Oplus, OverDrive, PDCharm, Pentium, Pentium Inside, skool, Sound Mark, The Journey Inside, Viiv Inside, vPro Inside, VTune, Xeon, and Xeon Inside are trademarks of Intel Corporation in the U.S. and other countries.

* Other names and brands may be claimed as the property of others.

Copyright © 2009, Intel Corporation. All rights reserved.

Microsoft product screen shot(s) reprinted with permission from Microsoft Corporation.