Intel® C++ Compiler Professional Edition 11.1 for Linux* Installation Guide and Release Notes

Document number: 321412-002US
20 October 2009

Table of Contents

1 Introduction 3
1.1 Change History 3
1.2 Product Contents 4
1.3 System Requirements 4
1.3.1 Red Hat Enterprise Linux* 3, SUSE LINUX Enterprise Server* 9 Support Deprecated 7
1.4 Documentation 7
1.5 Japanese Language Support 7
1.6 Technical Support 7
2 Installation 7
2.1.1 Silent Install 8
2.1.2 Eclipse* Integration Installation 8
2.1.3 Known Installation Issues 8
2.2 Installation Folders 9
2.3 Removal/Uninstall 10
3 Intel® C++ Compiler 11
3.1 Compatibility 11
3.2 New and Changed Features 11
3.3 New and Changed Compiler Options 11
3.3.1 –O0 no longer implies –mp 11
3.4 Other Changes 11
3.4.1 Optimization Reports Disabled by Default 11
3.4.2 Establishing the Compiler Environment 12

3.4.3 Instruction Set Default Changed to Require Intel® Streaming SIMD Extensions 2 (Intel® SSE2) 12
3.4.4 OpenMP* Libraries Default to “compat” 12
3.4.5 `mathf.h` No Longer Provided 12
3.4.6 Sampling-based Profile Guided Optimization Feature Removed 13
3.5 Using Source Checker in the Eclipse* IDE 13
3.6 Known Issues 13
3.6.1 TR1 System Headers 13
3.6.2 The default behavior for `KMP_AFFINITY` has changed 14
3.6.3 Fatal error from old version of ld 14
3.6.4 Certain Intel® AVX Architecture Instructions and Intrinsics Removed 14
4 Intel® Debugger (IDB) 14
4.1 Setting up the Java Runtime Environment 15
4.2 Starting the Debugger 15
4.3 Additional Documentation 15
4.4 Debugger Features 15
4.4.1 Main Features of IDB 15
4.4.2 New and Changed Features 15
4.5 Known Problems 16
4.5.1 Data Sharing Detection Issues 16
4.5.2 Signals Dialog not working 16
4.5.3 Resizing GUI 16
4.5.4 Kill Process 16
4.5.5 Decimal Floating Point Not Supported 16
4.5.6 `$cdir, $cwd` Directories 16
4.5.7 `info stack` Usage 17
4.5.8 `$stepg0` Default Value Changed 17
4.5.9 SIGTRAP error on some Linux* Systems 17
5 Eclipse Integration 17
5.1 Supplied Integrations 17
5.1.1 Eclipse 3.5 and CDT 6.0 17
5.1.2 Eclipse 3.4 and CDT 5.0 18
5.1.3 Integration notes 18

5.2 How to Install the Intel C++ Eclipse Product Extension in Your Eclipse Platform 18
5.2.1 Eclipse 3.5.0 and CDT 6.0.0 “Galileo” 18
5.2.2 Eclipse 3.4.0 and CDT 5.0.0 “Ganymede” 19
5.3 How to Obtain and Install Eclipse, CDT and a JRE 20
5.3.1 Eclipse 3.5.0 with CDT 6.0.0 20
5.3.2 Eclipse 3.4.0 and CDT 5.0.0 20
5.3.3 Installing JRE, Eclipse and CDT 20
5.4 Launching Eclipse for Development with the Intel C++ Compiler 20
5.5 Installing on Fedora* Systems 21
5.6 Selecting Compiler Versions 21
6 Intel® Integrated Performance Primitives 21
6.1 New and Changed Features 21
6.2 Known Limitations 22
6.3 Intel® IPP Cryptography Libraries are Available as a Separate Download 22
6.4 Intel® IPP Code Samples 22
7 Intel® Math Kernel Library 22
7.1 Changes in This Version 22
7.1.1 New features 22
7.1.2 Usability/Interface improvements 23
7.1.3 Performance improvements 23
7.2 Known Issues 25
7.3 Notices 25
7.4 Attributions 25
8 Intel® Threading Building Blocks 26
9 Disclaimer and Legal Information 26

1 Introduction

This document describes how to install the product, provides a summary of new and changed features and includes notes about features and problems not described in the product documentation.

1.1 Change History

This section highlights important changes in product updates. For a list of corrections to reported problems, please read Intel® Professional Edition Compilers 11.1 Fixes List.

Update 3

- Intel® Threading Building Blocks updated to 2.2 Update 1
- Corrections to reported problems

Update 2 (11.1.056)

- Ubuntu* 9.04 now supported
- Note added about non-RPM install forced on Fedora* 10.
- Note added about new options `-mkl` and `-xAVX`
- Note added about removal of certain Intel® AVX architecture instructions and intrinsics
- Full support for Eclipse* CDT 6.0 is now available
- Corrections to reported problems

Update 1 (11.1.046)

- Note added about change in behavior of `-O0`
- Partial support for Eclipse CDT 6.0 added
- Corrections to reported problems

1.2 Product Contents

*Intel® C++ Compiler Professional Edition 11.1 for Linux** includes the following components:

- Intel® C++ Compilers for building applications that run on IA-32, Intel® 64 and IA-64 architecture systems running the Linux* operating system
- Intel® Debugger
- Intel® Assembler for IA-64 architecture applications
- Intel® Integrated Performance Primitives 6.1 Update 2
- Intel® Math Kernel Library 10.2 Update 2
- Intel® Threading Building Blocks 2.2 Update 1
- Integration into the Eclipse* development environment
- On-disk documentation

1.3 System Requirements

For an explanation of architecture names, see http://software.intel.com/en-us/articles/intel-architecture-platform-terminology/

Requirements to develop IA-32 architecture applications

- A PC based on an IA-32 or Intel® 64 architecture processor supporting the Intel® Streaming SIMD Extensions 2 (Intel® SSE2) instructions (Intel® Pentium® 4 processor or later, or compatible non-Intel processor)
    - Development for a target different from the host may require optional library components to be installed from your Linux Distribution.
    - For the best experience, a multi-core or multi-processor system is recommended
- 1GB of RAM (2GB recommended)

- 3GB free disk space for all features
- One of the following Linux distributions (this is the list of distributions tested by Intel; other distributions may or may not work and are not recommended - please refer to the Technical Support section if you have questions):
    - Asianux* 3.0
    - Debian* 4.0
    - Fedora* 10
    - Red Hat Enterprise Linux* 3, 4, 5
    - SUSE LINUX Enterprise Server* 9, 10, 11
    - TurboLinux* 11
    - Ubuntu* 9.04
- Linux Developer tools component installed, including gcc, g++ and related tools
- binutils 2.17.50 or later
- Linux component compat-libstdc++ providing libstdc++.so.5
- If developing on an Intel® 64 architecture system, some Linux distributions may require installation of one or more of the following additional Linux components: ia32-libs, lib32gcc1, lib32stdc++6, libc6-dev-i386, gcc-multilib

Requirements to Develop Intel® 64 Architecture Applications

- A PC based on an Intel® 64 architecture processor (Intel® Pentium 4 processor or later, or compatible non-Intel processor)
    - For the best experience, a multi-core or multi-processor system is recommended
- 1GB of RAM (2GB recommended)
- 3GB free disk space for all features
- 100 MB of hard disk space for the virtual memory paging file. Be sure to use at least the minimum amount of virtual memory recommended for the installed distribution of Linux
- One of the following Linux distributions (this is the list of distributions tested by Intel; other distributions may or may not work and are not recommended - please refer to the Technical Support section if you have questions):
    - Asianux* 3.0
    - Debian* 4.0
    - Fedora* 10
    - Red Hat Enterprise Linux* 3, 4, 5
    - SGI ProPack* 5
    - SUSE LINUX Enterprise Server* 9, 10, 11
    - TurboLinux* 11
    - Ubuntu* 9.04
- Linux Developer tools component installed, including gcc, g++ and related tools
- binutils 2.17.50 or later
- Linux component compat-libstdc++ providing libstdc++.so.5
- Linux component containing 32-bit libraries (may be called ia32-libs)

Requirements to Develop IA-64 Architecture Applications

- A system based on an IA-64 architecture processor (Intel® Itanium®)
- 1GB of RAM (2 GB recommended).
- 3GB free disk space for all features
- One of the following Linux distributions (this is the list of distributions tested by Intel; other distributions may or may not work and are not recommended - please refer to the Technical Support section if you have questions):
    - Asianux* 3.0
    - Debian* 4.0
    - Red Hat Enterprise Linux* 3, 4, 5
    - SUSE LINUX Enterprise Server* 9, 10, 11
    - TurboLinux* 11
    - Ubuntu* 9.04
- Linux Developer tools component installed, including gcc, g++ and related tools
- binutils 2.17.50 or later
- Linux component compat-libstdc++ providing libstdc++.so.5

Additional Requirements to use the Graphical User Interface of the Intel® Debugger

- IA-32 architecture system or Intel® 64 architecture system
- Java* Runtime Environment (JRE) 5.0 (also called 1.5) or 6.0 (1.6)
    - A 32-bit JRE must be used on an IA-32 architecture system and a 64-bit JRE must be used on an Intel® 64 architecture system

Additional Requirements to use Eclipse Integration*

- IA-32 architecture system or Intel® 64 architecture system
- Eclipse* 3.5.x or 3.4.x
- Eclipse C/C++ Development Tools (CDT) 6.0.x or 5.0.x
- Java Run-Time Environment 5.0 (1.5) or 6.0 (1.6)

Notes

- The Intel compilers are tested with a number of different Linux distributions, with different versions of gcc. Some Linux distributions may contain header files different from those we have tested, which may cause problems. The version of glibc you use must be consistent with the version of gcc in use. For best results, use only the gcc versions as supplied with distributions listed above.
- Compiling very large source files (several thousands of lines) using advanced optimizations such as `-O3`, `-ipo` and `-openmp`, may require substantially larger amounts of RAM.
- The above lists of processor model names are not exhaustive - other processor models correctly supporting the same instruction set as those listed are expected to work. Please refer to the *Technical Support* section if you have questions regarding a specific processor model

- Some optimization options have restrictions regarding the processor type on which the application is run. Please see the documentation of these options for more information.

1.3.1 Red Hat Enterprise Linux* 3, SUSE LINUX Enterprise Server* 9 Support Deprecated

In a future major release of Intel C++ Compiler, support will be removed for installation and use on Red Hat Enterprise Linux 3 and SUSE LINUX Enterprise Server 9. Intel recommends migrating to a newer version of these operating systems.

1.4 Documentation

Product documentation can be found in the `Documentation` folder as shown under Installation Folders.

1.5 Japanese Language Support

Intel compilers provide support for Japanese language users. Error messages, visual development environment dialogs and some documentation are provided in Japanese in addition to English. By default, the language of error messages and dialogs matches that of your operating system language selection. Japanese-language documentation can be found in the `ja_JP` subdirectory for documentation and samples.

If you wish to use Japanese-language support on an English-language operating system, or English-language support on a Japanese-language operating system, you will find instructions at http://software.intel.com/en-us/articles/changing-language-setting-to-see-english-on-a-japanese-os-environment-or-vice-versa-on-linux/

1.6 Technical Support

Register your license at the Intel® Software Development Products Registration Center. Registration entitles you to free technical support, product updates and upgrades for the duration of the support term.

For information about how to find Technical Support, Product Updates, User Forums, FAQs, tips and tricks, and other support information, please visit:
http://www.intel.com/software/products/support/

Note: If your distributor provides technical support for this product, please contact them for support rather than Intel.

2 Installation

If you are installing the product for the first time, please be sure to have the product serial number available as you will be asked for it during installation. A valid license is required for installation and use.

If you received your product on DVD, mount the DVD, change the directory (`cd`) to the top-level directory of the mounted DVD and begin the installation using the command:

```
./install.sh
```

If you received the product as a downloadable file, first unpack it into a writeable directory of your choice using the command:

```
tar -xzvf name-of-downloaded-file
```

Then change the directory (`cd`) to the directory containing the unpacked files and begin the installation using the command:

```
./install.sh
```

Follow the prompts to complete installation.

2.1.1 Silent Install

For information on automated or “silent” install capability, please see http://software.intel.com/en-us/articles/intel-compilers-for-linux-version-111-silent-installation-guide/

2.1.2 Eclipse* Integration Installation

Please refer to the section below on Eclipse Integration

2.1.3 Known Installation Issues

- If you have enabled the Security-Enhanced Linux (SELinux) feature of your Linux distribution, you must change the `SELINUX` mode to `permissive` before installing the Intel C++ Compiler. Please see the documentation for your Linux distribution for details. After installation is complete, you may reset the `SELINUX` mode to its previous value.
- On some versions of Linux, auto-mounted devices do not have the "exec" permission and therefore running the installation script directly from the DVD will result in an error such as:

  ```
  bash: ./install.sh: /bin/bash: bad interpreter: Permission denied
  ```

  If you see this error, remount the DVD with exec permission, for example:

  ```
  mount /media/<dvd_label> -o remount,exec
  ```

  and then try the installation again.
- On Fedora* 10 systems, the compiler forces a non-RPM install because some versions of Fedora 10 contain a defective `rpm` utility that prevents the Intel® compiler from installing properly.
- The version 11.1 product is fully supported on Ubuntu 9.04 IA-32 and Intel® 64 architecture systems. Due to a restriction in the licensing software, however, it is not possible to use the Trial License feature when evaluating IA-32 components on an Intel® 64 architecture system with Ubuntu 9.04. Earlier versions of Ubuntu, not officially supported by this release of software, may have similar problems. This affects using a

Trial License only. Use of serial numbers, license files, floating licenses or other license manager operations, and off-line activation (with serial numbers) is not affected. If you need to evaluate IA-32 components of the version 11.1 product on an Intel® 64 architecture Ubuntu system, please visit the Intel Software Evaluation Center (http://www.intel.com/cd/software/products/asmo-na/eng/download/eval/) to obtain an evaluation serial number.

2.2 Installation Folders

The arrangement of installation folders is shown in the diagram below. Not all folders will be present in a given installation.

- <install-dir>/Compiler/11.1/*xxx*/
  - bin
    - ia32
    - intel64
    - ia64
  - include
    - ia32
    - intel64
    - ia64
  - perf_headers
  - substitute_headers
  - lib
    - ia32
    - intel64
    - ia64
  - eclipse_support
  - idb
    - eclipse_support
    - gui
    - ia32
    - ia64
    - intel64
    - lib
    - third_party
  - ipp
    - em64t
    - ia32
    - ia64
  - mkl
    - benchmarks
    - examples
    - include
    - interfaces

        - lib
        - tests
        - tools
    - tbb
        - bin
        - em64t
        - examples
        - ia32
        - include
        - itanium
        - lib
    - Documentation
    - man
    - Samples

Where <install-dir> is the installation directory (default for system-wide installation is /opt/intel) and *xxx* is the three-digit build number and the folders under bin, include and lib are used as follows:

- ia32: Files used to build applications that run on IA-32
- intel64: Files used to build applications that run on Intel® 64
- ia64: Files used to build applications that run on IA-64

If you have both the Intel C++ and Intel Fortran compilers installed, they will share folders for a given version.

2.3 Removal/Uninstall

Removing (uninstalling) the product should be done by the same user who installed it (root or a non-root user). If sudo was used to install, it must be used to uninstall as well. It is not possible to remove the compiler while leaving any of the performance library or Eclipse* integration components installed.

1. Open a terminal window and set default (cd) to any folder outside <install-dir>
2. Type the command: <install-dir>/bin/ia32/uninstall_cproc.sh (substitute intel64 or ia64 for ia32 as desired)
3. Follow the prompts
4. Repeat steps 2 and 3 to remove additional platforms or versions

If you have the same-numbered version of Intel® Fortran Compiler installed, it may also be removed. If you have added the Intel C++ Eclipse integration to an instance of Eclipse in your environment, you will need to update your Eclipse configuration by removing the Intel integration extension site from your Eclipse configuration.

3 Intel® C++ Compiler

This section summarizes changes, new features and late-breaking news about the Intel C++ Compiler.

3.1 Compatibility

In version 11.0, the IA-32 architecture default for code generation changed to assume that Intel® Streaming SIMD Extensions 2 (Intel® SSE2) instructions are supported by the processor on which the application is run. See below for more information.

3.2 New and Changed Features

Please refer to the compiler documentation for details

- Additional features from C++0x
- C++ lambda functions
- Language extensions for parallel execution
- Asynchronous I/O extensions
- Decimal floating point
- #pragma vector_nontemporal
- #pragma unroll_and_jam
- valarray implementation using IPP option

3.3 New and Changed Compiler Options

- -diag-enable sc-include
- -diag-enable sc-parallel
- -mkl[=lib]
- -xAVX

For a list of deprecated compiler options, see the Compiler Options section of the documentation.

3.3.1 –O0 no longer implies –mp

In version 11.1, the `-O0` option for disabling optimizations no longer implies `-mp` for maximizing floating-point precision. The `-mp` switch is deprecated; Intel recommends using an explicit `-fp-model` option for applications that are sensitive to floating-point precision changes.

3.4 Other Changes

3.4.1 Optimization Reports Disabled by Default

As of version 11.1, the compiler no longer issues, by default, optimization report messages regarding vectorization, automatic parallelization and OpenMP threaded loops. If you wish to see these messages you must request them by specifying `-diag-enable vec`, `-diag-enable par` and/or `-diag-enable openmp`, or by using `-vec-report`, `-par-report` and/or `-openmp-report`.

Also, as of version 11.1, optimization report messages are sent to `stderr` and not `stdout`.

3.4.2 Establishing the Compiler Environment

The `iccvars.sh` (`iccvars.csh`) script, used to set up the command-line build environment, has changed. In previous versions, you chose the target platform by selecting either the `cc` or `cce` directory root. In version 11.x, there is one version of these scripts and they now take an argument to select the target platform.

The command takes the form:

`source <install-dir>/Compiler/11.1/`*`xxx`*`/bin/iccvars.sh `*`argument`*

Where `<install-dir>` is the installation directory (default for system-wide installation is `/opt/intel`) and *`xxx`* is the build number and *`argument`* is one of `ia32`, `intel64`, `ia64` as described above under Installation Folders. Establishing the compiler environment also establishes the Intel® Debugger (idb) environment.

3.4.3 Instruction Set Default Changed to Require Intel® Streaming SIMD Extensions 2 (Intel® SSE2)

When compiling for the IA-32 architecture, `-msse2` (formerly `-xW`) is the default as of version 11.0. Programs built with `-msse2` in effect require that they be run on a processor that supports the Intel® Streaming SIMD Extensions 2 (Intel® SSE2), such as the Intel® Pentium® 4 processor and some non-Intel processors. No run-time check is made to ensure compatibility – if the program is run on an unsupported processor, an invalid instruction fault may occur. Note that this may change floating point results since the Intel® SSE instructions will be used instead of the x87 instructions and therefore computations will be done in the declared precision rather than sometimes a higher precision.

All Intel® 64 architecture processors support Intel® SSE2.

To specify the older default of generic IA-32, specify `-mia32`

3.4.4 OpenMP* Libraries Default to "compat"

In version 10.1, a new set of OpenMP libraries was added that allowed applications to use OpenMP* code from both Intel and Microsoft compilers. These "compatibility" libraries can provide higher performance than the older "legacy" libraries. In version 11.x, the compatibility libraries are used by default for OpenMP applications, equivalent to `-openmp-lib compat`. If you wish to use the older libraries, specify `-openmp-lib legacy`

The "legacy" libraries will be removed in a future release of the Intel compilers.

3.4.5 `mathf.h` No Longer Provided

The header file `mathf.h`, used to define single-precision math library functions, has been removed from the product. If you were using this header file, please use `mathimf.h` instead.

3.4.6 Sampling-based Profile Guided Optimization Feature Removed

The hardware sampling-based Profile-Guided Optimization feature is no longer provided. The `-prof-gen-sampling` and `-ssp` compiler options and the `profrun` and `pronto_tool` executables have been removed. Instrumented Profile-Guided Optimization is still supported.

3.5 Using Source Checker in the Eclipse* IDE

When Source Checker (formerly Static Verifier) support is enabled within the IDE, the customary final build target (e.g. an executable image) is not created. As such, we recommend that a separate "Source Checking" configuration be created, by cloning the existing Debug (development) configuration, for use when static verification is desired.

- Open the property pages for the project and select "C/C++ Build".
- Click the "Manage" button
- In the "Manage" dialog, click the "New" button to open the "Create configuration" dialog.
- Supply a name for the new configuration in the "Name" box.
- Supply a "Description" for the configuration if you want (optional).
- You can choose to "Copy settings from" a "Default configuration" or an "Existing configuration" by clicking the appropriate radio button and then selecting a configuration from the corresponding drop down menu.
- Click "O.K." to close the "Create configuration" dialog.
- Click "O.K." to close the "Manage" dialog (with your new configuration name selected).
- The property pages will now be displaying the settings for your new configuration and it is now the active build configuration.
- Navigate to the Intel compiler's Compilation Diagnostics properties. Use the "Level of Source Code Analysis", “Level of Source Code Parallelization Analysis” and "Analyze Included Files" properties to control source code analysis.

3.6 Known Issues

3.6.1 TR1 System Headers

If you are using the TR1 (C++ Library Technical Report 1) system headers on a system with g++ version 4.3 or later installed, the Intel C/C++ compiler will give errors when it tries to compile the <type_traits> header file. This is because the Intel C/C++ compiler does not yet support the C++0x feature called variadic templates. You will see these types of compilation errors:

```
../include/c++/4.3.0/tr1_impl/type_traits(170): error: expected an
identifier
   template<typename _Res, typename... _ArgTypes>

                                       ^

include/c++/4.3.0/tr1_impl/type_traits(171): error: expected a ")"
    struct __is_function_helper<_Res(_ArgTypes...)>
```

There is no workaround, other than not using these headers or using an older version of the g++ compiler.

3.6.2 The default behavior for KMP_AFFINITY has changed

The thread affinity type of the KMP_AFFINITY environment variable defaults to none (KMP_AFFINITY=none). The behavior for KMP_AFFINITY=none was changed in 10.1.015 or later, and in all 11.x compilers, such that the initialization thread creates a "full mask" of all the threads on the machine, and every thread binds to this mask at startup time. It was subsequently found that this change may interfere with other platform affinity mechanisms, for example, dplace() on SGI Altix machines. To resolve this issue, a new affinity type disabled was introduced in compiler 10.1.018, and in all 11.1 compilers (KMP_AFFINITY=disabled). Setting KMP_AFFINITY=disabled will prevent the OpenMP runtime library from making any affinity-related system calls.

3.6.3 Fatal error from old version of ld

In some circumstances, linking of an application with the version 11.x compiler will fail with an ld internal error similar to the following:

```
ld: BFD 2.15.92.0.2 20040927 internal error, aborting at
../../bfd/reloc.c line 444 in bfd_get_reloc_size
ld: Please report this bug.
```

To resolve this, install a more recent version of binutils. 2.17.50 is the recommended minimum version.

3.6.4 Certain Intel® AVX Architecture Instructions and Intrinsics Removed

The compiler does not support the VPERMIL2PD and VPERMIL2PS instructions in the assembler, and does not support the corresponding intrinsics _mm256_permute2_pd, _mm_permute2_pd, _mm256_permute2_ps, and _mm_permute2_ps. These instructions and intrinsics were removed from the Intel® AVX architecture but not from the compiler documentation. For more information, please see http://software.intel.com/en-us/blogs/2009/01/29/recent-intelr-avx-architectural-changes/

4 Intel® Debugger (IDB)

The following notes refer to the Graphical User Interface (GUI) available for the Intel® Debugger (IDB) when running on IA-32 and Intel® 64 architecture systems. In this version, the idb command invokes the GUI – to get the command-line interface, use idbc.

On IA-64 architecture systems, the GUI is not available and the idb command invokes the command-line interface.

4.1 Setting up the Java Runtime Environment

The Intel® IDB Debugger graphical environment is a Java application and requires a Java Runtime Environment (JRE) to execute. The debugger will run with a version 5.0 (also called 1.5) or 6.0 (1.6) JRE.

Install the JRE according to the JRE provider's instructions.

Finally you need to export the path to the JRE as follows:

```
export PATH=<path_to_JRE_bin_dir>:$PATH
```

4.2 Starting the Debugger

To start the debugger, first make sure that the compiler environment has been established as described at Establishing the Compiler Environment. Then use the command:

```
idb
```

or

```
idbc
```

as desired.

Once the GUI is started and you see the console window, you're ready to start the debugging session.

Note: Make sure that the executable you want to debug is built with debug info and is an executable file. Change permissions if required, e.g. `chmod +x <application_bin_file>`

4.3 Additional Documentation

Online help titled *Intel® Compilers / Intel® Debugger Online Help* is accessible from the debugger graphical user interface as `Help > Help Contents`.

Context-sensitive help is also available in several debugger dialogs where a `Help` button is displayed.

4.4 Debugger Features

4.4.1 Main Features of IDB

The debugger supports all features of the command line version of the Intel® IDB Debugger. Debugger functions can be called from within the debugger GUI or the GUI-command line. Please refer to the Known Limitations when using the graphical environment.

4.4.2 New and Changed Features

- Debugger GUI for IA-32 and Intel® 64 architectures
- Parallel Execution Debug Support
- Session Concept
- Bitfield editor

- SIMD registers window
- OpenMP support
    - Information windows for Tasks, Barriers, Taskwaits, Locks, Teams and Task Spawn Tree
    - Data sharing events and reentrancy call detection
    - Debugging serialized code without recompilation
- Internationalization support

4.5 Known Problems

4.5.1 Data Sharing Detection Issues

- When the `Stop On Event` icon is deactivated, or the `Parallel > Stop on Event` menu item is unchecked, data sharing events are not collected in the Data Sharing Events window. If you stop the debugger and open the Data Sharing Events window, you will see only the last event.
- When the `Data Sharing Events` window is closed and reopened, a new Analysis Run Node will be displayed that duplicates recent events.
- When the `Data Sharing Events` window is closed during data sharing detection, only the last event is displayed when the window is reopened after detection.

4.5.2 Signals Dialog not working

The Signals dialog accessible via the GUI dialog Debug / Signal Handling or the shortcut Ctrl+S is not working correctly. Please refer to the Intel® Debugger (IDB) Manual for use of the signals command line commands instead.

4.5.3 Resizing GUI

If the debugger GUI window is reduced in size, some windows may fully disappear. Enlarge the window and the hidden windows will appear again.

4.5.4 Kill Process

The 'Kill Focused Process' command from the Debug menu does not work when the debugger is running. Stop the debugger first and then kill the process.

4.5.5 Decimal Floating Point Not Supported

The Intel® Debugger does not support decimal floating point data types, supported in some C++ compilers. The debugger will display such values as if they were arrays of characters.

4.5.6 `$cdir, $cwd` Directories

`$cdir` is the compilation directory (if recorded). This is supported in that the directory is set; but `$cdir` is not itself supported as a symbol.

`$cwd` is the current working directory. Neither the semantics nor the symbol are supported.

The difference between `$cwd` and '.' is that `$cwd` tracks the current working directory as it changes during a debug session. '.' is immediately expanded to the current directory at the time an entry to the source path is added.

4.5.7 `info stack` Usage

The debugger command `info stack` does not currently support negative frame counts in the optional syntax below:

```
info stack [num]
```

A positive frame count num will print the innermost num frames. A negative or zero count will print no frames rather than the outermost num frames.

4.5.8 `$stepg0` Default Value Changed

The debugger variable $stepg0 changed default to a value of 0. With the value "0" the debugger will step over code without debug information if you do a "step" command. Set the debugger variable to 1 to be compatible with previous debugger versions as follows:

```
(idb) set $stepg0 = 1
```

4.5.9 SIGTRAP error on some Linux* Systems

On some rare cases with special Linux kernels a SIGTRAP error may occur when the debugger stops at a breakpoint and you continue debugging. As a workaround you can define the SIGTRAP signal as follows on command line:

```
(idb) handle SIGTRAP nopass noprint nostop
SIGTRAP is used by the debugger.
SIGTRAP       No        No      No              Trace/breakpoint trap
(idb)
```

4.5.10 idb GUI cannot be used to debug MPI processes

The idb GUI cannot be used to debug MPI processes. The command line interface (idbc) can be used for this purpose.

Eclipse Integration

The Intel C++ Compiler for the IA-32 and Intel® 64 architectures installs an Eclipse feature and associated plugins (the Intel C++ Eclipse Product Extension) which provide support for the Intel C++ compiler when added as an Eclipse product extension site to an existing instance of the Eclipse* Integrated Development Environment (IDE). With this feature, you will be able to use the Intel C++ compiler from within the Eclipse integrated development environment to develop your applications.

4.6 Supplied Integrations

4.6.1 Eclipse 3.5 and CDT 6.0

The Intel feature provided in the directory

```
<install-dir>/eclipse_support/cdt6.0/eclipse
```

supports and requires Eclipse Platform version 3.5.x, Eclipse C/C++ Development Tools (CDT) version 6.0.0 or later and a functional Java Runtime Environment (JRE) (minimum version 5.0 (also called 1.5), recommended version 6.0).

4.6.2 Eclipse 3.4 and CDT 5.0

The Intel feature provided in the directory

```
<install-dir>/eclipse_support/cdt5.0/eclipse
```

supports and requires Eclipse Platform version 3.4.x, Eclipse C/C++ Development Tools (CDT) version 5.0.0 or later and a functional Java Runtime Environment (JRE) (minimum version 5.0 (also called 1.5), recommended version 6.0).

4.6.3 Integration notes

Note that the Eclipse Platform versions 3.5 and 3.4 are not currently available for the IA-64 architecture. The compiler kit includes an Eclipse integration for that architecture should the platform be released at a later date.

If you already have the proper versions of Eclipse, CDT and a functional JRE installed and configured in your environment, then you can add the Intel C++ Eclipse Product Extension to your Eclipse Platform, as described in the section, below, entitled How to Install the Intel C++ Eclipse Product Extension in Your Eclipse Platform. Otherwise, you will first need to obtain and install Eclipse, CDT and a JRE, as described in the section, below, entitled How to Obtain and Install Eclipse, CDT and a JRE and then install the Intel C++ Eclipse Product Extension.

4.7 How to Install the Intel C++ Eclipse Product Extension in Your Eclipse Platform

To add the Intel C++ product extension to your existing Eclipse configuration, follow these steps, from within Eclipse.

4.7.1 Eclipse 3.5.0 and CDT 6.0.0 “Galileo”

Open the "Available Software" page by selecting: `Help > Install New Software...` Click on the "Add..." button. Select "Local...". A directory browser will open. Browse to select the `eclipse` directory in your Intel C++ compiler installation. For example, if you installed the compiler as root to the default directory, you would browse to `/opt/intel/Compiler/11.1/uuu/eclipse_support/cdt6.0/eclipse`. Select “OK” to close the directory browser. Then select "OK" to close the “Add Site” dialog. Select the two boxes for the Intel C++ integration: there will be one box for “Intel® C++ Compiler Documentation” and a second box for “Intel® C++ Compiler Professional 11.1 for Linux*”. **Note:** The Intel features will not be visible if you have `Group items by category` set – unset this option to view the Intel features.

Click the “Install” button. An “Install” dialog will open which gives you a chance to review and confirm you want to install the checked items. Click “Next”. You will now be asked to accept the license agreement. Accept the license agreement and click “Finish”. The installation of the Intel support will proceed.

When asked to restart Eclipse, select “Yes”. When Eclipse restarts, you will be able to create and work with CDT projects that use the Intel C++ compiler. See the Intel C++ Compiler documentation for more information. You can find the Intel C++ documentation under `Help > Help Contents > Intel C++ Compiler Users Guide`. If you also installed the Intel® Debugger (idb) along with the idb Eclipse product extension, and would like to use idb within Eclipse, you should add the idb product extension site to your Eclipse configuration in the same way. For example, if you installed the kit as root to the default directory, you would find the idb Eclipse product extension site at `/opt/intel/Compiler/11.1/uuu/idb/eclipse_support/cdt6.0/eclipse`.

4.7.2 Eclipse 3.4.0 and CDT 5.0.0 “Ganymede”

Open the "Software Updates and Add-ons" page by selecting:

```
Help > Software Updates...
```

Select the "Available Software" tab.

Select "Add Site..." and then "Local...". A directory browser will open. Browse to select the `eclipse` directory in your Intel C++ compiler installation. For example, if you installed the compiler as root to the default directory, you would browse to `/opt/intel/Compiler/11.1/uuu/eclipse_support/cdt5.0/eclipse`.

Select "OK" to close the directory browser. Then select "OK" to close the “Add Site” dialog. Select the two boxes for the Intel C++ integration: there will be one box for “Intel® C++ Compiler Documentation” and a second box for “Intel® C++ Compiler 11.1 for Linux*”. **Note:** The Intel features will not be visible if you have `Group items by category` set – unset this option to view the Intel features.

Click the “Install” button. An “Install” dialog will open which gives you a chance to review and confirm you want to install the checked items. Click “Next”. You will now be asked to accept the license agreement. Accept the license agreement and click “Finish”. The installation of the Intel support will proceed.

When asked to restart Eclipse, select “Yes”. When Eclipse restarts, you will be able to create and work with CDT projects that use the Intel C++ compiler. See the Intel C++ Compiler documentation for more information. You can find the Intel C++ documentation under `Help > Help Contents > Intel C++ Compiler User and Reference Guides`.

If you also installed the Intel® Debugger (idb) along with the idb Eclipse product extension, and would like to use idb within Eclipse, you should add the idb product extension site to your Eclipse configuration in the same way. For example, if you installed the kit as root to the default directory, you would find the idb Eclipse product extension site at `/opt/intel/Compiler/11.1/uuu/idb/eclipse_support/cdt5.0/eclipse`.

4.8 How to Obtain and Install Eclipse, CDT and a JRE

Eclipse is a Java application and therefore requires a Java Runtime Environment (JRE) to execute. The 3.4.0 version of the Eclipse platform will run with a minimum version 5.0 JRE. Intel recommends using a 6.0 (1.6) JRE. The choice of a JRE is dependent on your operating environment (machine architecture, operating system, etc.) and there are many JRE's available to choose from.

4.8.1 Eclipse 3.5.0 with CDT 6.0.0

A package containing both Eclipse 3.5.0 and CDT 6.0.0 is available from:

```
http://www.eclipse.org/downloads/packages/eclipse-ide-cc-
developers/galileor
```

Choose either the Linux 32bit or Linux 64bit download as desired.

4.8.2 Eclipse 3.4.0 and CDT 5.0.0

A package containing both Eclipse 3.4.0 and CDT 5.0.0 is available from:

```
http://www.eclipse.org/downloads/packages/release/ganymede/r
```

Find “Eclipse IDE for C/C++ Developers” and choose either the Linux 32bit or Linux 64bit download as desired.

4.8.3 Installing JRE, Eclipse and CDT

Once you have downloaded the appropriate files for Eclipse, CDT, and a JRE, you can install them as follows:

1. Install your chosen JRE according to the JRE provider's instructions.
2. Create a directory where you would like to install Eclipse and `cd` to this directory. This directory will be referred to as `<eclipse-install-dir>`
3. Copy the Eclipse package binary `.tgz` file to the `<eclipse-install-dir>` directory.
4. Expand the `.tgz` file.
5. Start `eclipse`

You are now ready to add the Intel C++ product extension to your Eclipse configuration as described in the section, *How to Install the Intel C++ Eclipse Product Extension in Your Eclipse Platform*. If you need help with launching Eclipse for the first time, please read the next section.

4.9 Launching Eclipse for Development with the Intel C++ Compiler

If you have not already set your `LANG` environment variable, you will need to do so. For example,

```
setenv LANG en_US
```

Setup Intel C++ compiler related environment variables by executing the `iccvars.csh` (or `.sh`) script prior to starting Eclipse:

`source <install-dir>/bin/iccvars.csh arch_arg` (where "arch_arg" is one of "ia32" or "intel64").

Since Eclipse requires a JRE to execute, you must ensure that an appropriate JRE is available to Eclipse prior to its invocation. You can set the `PATH` environment variable to the full path of the folder of the `java` file from the JRE installed on your system or reference the full path of the java executable from the JRE installed on your system in the `-vm` parameter of the Eclipse command, e.g.:

```
eclipse -vm /JRE folder/bin/java
```

Invoke the Eclipse executable directly from the directory where it has been installed. For example:

```
<eclipse-install-dir>/eclipse/eclipse
```

4.10 Installing on Fedora* Systems

If the Intel C++ Compiler for Linux is installed on an IA-32 or Intel® 64 architecture Fedora* system as a "local" installation, i.e. not installed as root, the installation may fail to properly execute the Eclipse graphical user interfaces to the compiler or debugger. The failure mechanism will typically be displayed as a `JVM Terminated` error. The error condition can also occur if the software is installed from the root account at the system level, but executed by less privileged user accounts.

The cause for this failure is that a more granular level of security has been implemented on Fedora, but this new security capability can adversely affect access to system resources, such as dynamic libraries. This new *SELinux* security capability may require adjustment by your system administrator in order for the compiler installation to work for regular users.

4.11 Selecting Compiler Versions

For Eclipse projects you can select among the installed versions of the Intel C++ Compiler. On IA-32 architecture systems, the supported Intel compiler versions are 9.1, 10.0, 10.1, 11.0 and 11.1. On Intel® 64 architecture systems, only compiler versions 11.0 and 11.1 are supported.

5 Intel® Integrated Performance Primitives

This section summarizes changes, new features and late-breaking news about the Intel® Integrated Performance Primitives (Intel® IPP) as part of Intel C++ Compiler Professional Edition. For detailed information about IPP see the following links:

- **New features**: see the information below and visit the main Intel IPP product page on the Intel web site at: http://software.intel.com/en-us/intel-ipp.
- **Documentation, help, and samples**: see the documentation links on the IPP product page at: http://software.intel.com/en-us/intel-ipp.

5.1 4.1 New and Changed Features

- Prebuilt library binaries are now included with the data compression samples (bzip2, zlib, and gzip) making it even easier to quickly utilize the IPP library as part of your data compression applications.
- The ippiDemo application has been updated to include additional demonstration features, especially for comparing optimized performance versus non-optimized performance. Please see the ippiDemo readme file for more information.
- Support for the Advanced Encryption Standard (AES) instructions that are part of the SSE instructions on the Intel 32nm (Westmere code name)-based processors. These instructions enable the implementation of fast and secure data encryption and decryption algorithms.
- Data compression performance improvements for the Intel®-64 architecture resulting in significant speed gains for the ZLIB Inflate algorithm.

5.2 4.2 Known Limitations

- For a list of bug fixes, known issues, and limitations please see the following knowledge base article: http://software.intel.com/en-us/articles/intel-ipp-library-61-fixes-list/.

5.3 Intel® IPP Cryptography Libraries are Available as a Separate Download

The Intel® IPP cryptography libraries are available as a separate download. For download and installation instructions, please read
http://software.intel.com/en-us/articles/download-ipp-cryptography-libraries/

5.4 Intel® IPP Code Samples

The Intel® IPP code samples are organized into downloadable packages for Windows*, Linux* and Mac OS* at
http://software.intel.com/en-us/articles/intel-integrated-performance-primitives-code-samples/

The samples include source code for audio/video codecs, image processing and media player applications, and for calling functions from C++, C# and Java*. Instructions on how to build the sample are described in a readme file that comes with the installation package for each sample.

6 Intel® Math Kernel Library

This section summarizes changes, new features and late-breaking news about the Intel® Math Kernel Library as part of Intel C++ Compiler Professional Edition.

6.1 Changes in This Version

6.1.1 New features

- LAPACK 3.2

    - 238 new LAPACK functions
    - Extra Precise Iterative Refinement
    - Non-Negative Diagonals from Householder QR factorization
    - High Performance QR and Householder Reflections on Low-Profile Matrices
    - New fast and accurate Jacobi SVD
    - Routines for Rectangular Full Packed format
    - Pivoted Cholesky
    - Mixed precision iterative refinement (Cholesky)
    - More robust DQDS algorithm
- Introduced implementation of the DZGEMM Extended BLAS function (as described at http://www.netlib.org/blas/blast-forum/). See the description of the `*gemm` family of functions in the BLAS section of the reference manual.
- PARDISO now supports real and complex, single precision data

6.1.2 Usability/Interface improvements

- Sparse matrix format conversion routines which convert between the following formats:
    - CSR (3-array variation) ↔ CSC (3-array variation)
    - CSR (3-array variation) ↔ diagonal format
    - CSR (3-array variation) ↔ skyline
- Fortran95 BLAS and LAPACK compiled module files (.mod) are now included
    - Modules are pre-built with the Intel Fortran Compiler and are located in the include directory (see Intel® MKL User's Guide for full path)
    - Source is still available for use with other compilers
    - Documentation for these interfaces can be found in the Intel® MKL User's Guide
- The FFTW3 interface is now integrated directly into the main libraries
    - Source code is still available to create wrappers for use with compilers not compatible with the default Intel® Fortran compiler convention for name decoration
    - See Appendix G of the Reference Manual for information
- DFTI_DESCRIPTOR_HANDLE now represents a true type name and can now be referenced as a type in user programs
- Added parameter to Jacobi matrix calculation routine in the optimization solver domain to allow access to user data (see the description of the djacobix function in the reference manual for more information)
- Added an interface mapping calls to single precision BLAS functions in Intel® MKL (functions with "s" or "c" initial letter) to 64-bit floating point precision functions has been added on 64-bit architectures (See "sp2dp" in the Intel® MKL User Guide for more information)
- Compatibility libraries (also known as "dummy" libraries) have been removed from this version of the library

6.1.3 Performance improvements

- Further threading in BLAS level 1 and 2 functions for Intel® 64 architecture
    - Level 1 functions (vector-vector): (CS,ZD,S,D)ROT, (C,Z,S,D)COPY, and (C,Z,S,D)SWAP
        - Increase in performance by up to 1.7-4.7 times over version 10.1 Update 1 on 4-core Intel® Core™ i7 processor depending on data location in cache
        - Increase in performance by up to 14-130 times over version 10.1 Update 1 on 24-core Intel® Xeon® processor 7400 series system, depending on data location in cache
    - Level 2 functions (matrix-vector): (C,Z,S,D)TRMV, (S,D)SYMV, (S,D)SYR, and (S,D)SYR2
        - Increase in performance by up to 1.9-2.9 times over version 10.1 Update 1 on 4-core Intel® Core™ i7 processor, depending on data location in cache
        - Increase in performance by up to 16-40 times over version 10.1 Update 1 on 24-core Intel® Xeon® processor 7400 series system, depending on data location in cache
- Introduced recursive algorithm in 32-bit sequential version of DSYRK for up to 20% performance improvement on Intel® Core™ i7 processors and Intel® Xeon® processors in 5300, 5400, and 7400 series.
- Improved LU factorization (DGETRF) by 25% over Intel MKL 10.1 Update 1 for large sizes on the Intel® Xeon® 7460 Processor; small sizes are also dramatically improved
- BLAS *TBMV/*TBSV functions now use level 1 BLAS functions to improve performance by up to 3% on Intel® Core™ i7 processors and up to 10% on Intel® Core™2 processor 5300 and 5400 series.
- Improved threading algorithms to increase DGEMM performance
    - up to 7% improvement on 8 threads and up to 50% on 3,5,7 threads on the Intel® Core™ i7 processor
    - up to 50% improvement on 3 threads on Intel® Xeon® processor 7400 series.
- Threaded 1D complex-to-complex FFTs for non-prime sizes
- New algorithms for 3D complex-to-complex transforms deliver better performance for small sizes (up to 64x64x64) on 1 or 2 threads
- Implemented high-level parallelization of out-of-core (OOC) PARDISO when operating on symmetric positive definite matrices.
- Reduced memory use by PARDISO for both in-core and out-of-core on all matrix types
- PARDISO OOC now uses less than half the memory previously used in Intel MKL 10.1 for real symmetric, complex Hermitian, or complex symmetric matrices
- Parallelized Reordering and Symbolic factorization stage in PARDISO/DSS
- Up to 2 times better performance (30% improvement on average) on Intel® Core® i7 and Intel® Core™2 processors for the following VML functions: v(s,d)Round, v(s,d)Inv, v(s,d)Div, v(s,d)Sqrt, v(s,d)Exp, v(s,d)Ln, v(s,d)Atan, v(s,d)Atan2
- Optimized versions of the following functions available for Intel® Advanced Vector Extensions (Intel® AVX)
    - BLAS: DGEMM

- FFTs
- VML: exp, log, and pow
- See important information in the Intel® MKL User's Guide regarding the `mkl_enable_instructions()` function for access to these functions

6.2 Known Issues

A full list of the known limitations of this release can be found in the Knowledge Base for the Intel® MKL at http://software.intel.com/en-us/articles/known-limitations-in-intel-mkl-10-2

6.3 Notices

The following change is planned for future versions of Intel MKL. Please contact Technical Support if you have concerns:

- Content in the libraries containing `solver` in the filenames will be moved to the core library in a future version of Intel MKL. These `solver` libraries will then be removed.

6.4 Attributions

As referenced in the End User License Agreement, attribution requires, at a minimum, prominently displaying the full Intel product name (e.g. "Intel® Math Kernel Library") and providing a link/URL to the Intel® MKL homepage (www.intel.com/software/products/mkl) in both the product documentation and website.

The original versions of the BLAS from which that part of Intel® MKL was derived can be obtained from http://www.netlib.org/blas/index.html.

The original versions of LAPACK from which that part of Intel® MKL was derived can be obtained from http://www.netlib.org/lapack/index.html. The authors of LAPACK are E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. Our FORTRAN 90/95 interfaces to LAPACK are similar to those in the LAPACK95 package at http://www.netlib.org/lapack95/index.html. All interfaces are provided for pure procedures.

The original versions of ScaLAPACK from which that part of Intel® MKL was derived can be obtained from http://www.netlib.org/scalapack/index.html. The authors of ScaLAPACK are L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley.

PARDISO in Intel® MKL is compliant with the 3.2 release of PARDISO that is freely distributed by the University of Basel. It can be obtained at http://www.pardiso-project.org.

Some FFT functions in this release of Intel® MKL have been generated by the SPIRAL software generation system (http://www.spiral.net/) under license from Carnegie Mellon University. Some FFT functions in this release of the Intel® MKL DFTI have been generated by the UHFFT software generation system under license from University of Houston. The Authors of SPIRAL are Markus Puschel, Jose Moura, Jeremy Johnson, David Padua, Manuela Veloso, Bryan

Singer, Jianxin Xiong, Franz Franchetti, Aca Gacic, Yevgen Voronenko, Kang Chen, Robert W. Johnson, and Nick Rizzolo.

7 Intel® Threading Building Blocks

This section summarizes changes, new features and late-breaking news about Intel® Threading Building Blocks (Intel® TBB).

- Unhandled exceptions in the user code executed in the context of TBB algorithms or containers may lead to segmentation faults when Intel(R) C++ Compiler 10.x is used with glibc 2.3.2, 2.3.3, or 2.3.4.
- To allow more accurate results to be obtained with Intel® Thread Checker or Intel® Thread Profiler, download the latest update releases of these products before using them with Intel® Threading Building Blocks.
- If you are using Intel® Threading Building Blocks and OpenMP* constructs mixed together in rapid succession in the same program, and you are using Intel compilers for your OpenMP* code, set KMP_BLOCKTIME to a small value (e.g., 20 milliseconds to improve performance. This setting can also be made within your OpenMP* code via the kmp_set_blocktime() library call. See the compiler OpenMP* documentation for more details on KMP_BLOCKTIME and kmp_set_blocktime().
- In general, non-debug ("release") builds of applications or examples should link against the non-debug versions of the Intel® Threading Building Blocks libraries, and debug builds should link against the debug versions of these libraries. See the Tutorial in the product documentation sub-directory for more details on debug vs. release libraries.
- When using Ubuntu* 7.04 in 64-bit mode, compilations can fail with error messages saying that "`::system' has not been declared". These failures can be worked around by removing libpthread-dev from the system. See the following link for more details: https://bugs.launchpad.net/ubuntu/+source/gcc-4.1/+bug/77559

8 Disclaimer and Legal Information

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.

Parts of this product were built using third party libraries. Pursuant to the licenses ruling these libraries, Intel makes them available to users of this product. The libraries can be downloaded from Intel Software Development Products Knowledge Base article http://software.intel.com/en-us/articles/open-source-downloads/ . Please note that download of these libraries is not required to use the product.

MPEG-1, MPEG-2, MPEG-4, H.263, H.264, MP3, DV SD/25/50/100, VC-1, G.722.1, G.723.1A, G.726, G.728, G.729, GSM/AMR, GSM/FR, JPEG, JPEG 2000, Aurora, TwinVQ, AC3 and AAC are international standards promoted by ISO, IEC, ITU, SMPTE, ETSI and other organizations. Implementations of these standards or the standard enabled platforms may require licenses from various entities, including Intel Corporation.

Celeron, Centrino, Intel, Intel logo, Intel386, Intel486, Intel Atom, Intel Core, Itanium, MMX, Pentium, VTune, and Xeon are trademarks of Intel Corporation in the U.S. and other countries.

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