Intel® C++ Compiler Professional Edition 11.0 for Mac OS* X Installation Guide and Release Notes

Document number: 321414-001US
5 March 2009

1   Introduction

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

Note: All features and changes described in these Release Notes are subject to change or withdrawal before product release.

1.1   Product Contents

Intel® C++ Compiler Professional Edition 11.0 for Mac OS X* includes the following components:

- Intel® C++ Compilers for building applications that run on Intel-based Mac systems running the Mac OS* X operating system
- Intel® Debugger
- Intel® Integrated Performance Primitives
- Intel® Math Kernel Library
- Intel® Threading Building Blocks
- Integration into the Xcode* development environment
- On-disk documentation

1.2   Change History

The following section highlights new features and changes since the initial version 11.0 release. Update numbers may not represent a released update - the indicated and later updates have these changes.

- 11.0.060
  - Corrections to reported problems
- 11.0.059
  - Corrections to reported problems
- 11.0.056
  - Corrections to reported problems
- 11.0.054

Intel® C++ Compiler Professional Edition
11.0 for Mac OS* X Installation Guide and Release Notes

o   Initial release

1.3   System Requirements

- An Intel®-based Apple* Mac* system
- 512MB RAM minimum, 1GB RAM recommended
- 1GB free disk space
- Mac OS* X 10.4.11 or 10.5.5
- Mac OS* X Developer Tools including Xcode* 2.5 or 3.1.1
- gcc* 4

Note: Advanced optimization options or very large programs may require additional resources such as memory or disk space.

1.4   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 will be using Xcode*, please make sure that a supported version of Xcode is installed. If you install a new version of Xcode in the future, you must reinstall the Intel C++ Compiler afterwards.

If you received the compiler product on DVD, insert the DVD. Locate the disk image file (`m_cproc_p_11.0.xxx.dmg`) on the DVD and double-click. If you received the compiler product as a download, double-click the downloaded file, which will have a name of the form `m_cproc_p_11.0.xxx.dmg`.

Follow the prompts to complete installation.

1.5   Installation Folders

The 11.0 product installs into a different arrangement of folders than in previous versions. The new arrangement is shown in the diagram below. Not all folders will be present in a given installation.

- `<root>/intel/Compiler/11.0/xxx/`
    - `bin`
        - `ia32`
        - `intel64`
    - `include`
        - `ia32`
        - `intel64`
    - `lib`
    - `perf_headers`
    - `Frameworks`
        - `ipp`

Intel® C++ Compiler Professional Edition
11.0 for Mac OS* X Installation Guide and Release Notes

- mkl
- tbb
  - o Documentation
    - compiler_c
    - idb
    - ipp
    - mkl
    - tbb
  - o man
  - o Samples

Where `<root>` is `/opt` by default, *xxx* is the three-digit update number and the folders under `bin, include` and `lib` are used as follows:

- `ia32`: Compilers that build applications that run on a 32-bit Intel-based Mac OS* X system
- `intel64`: Compilers that build applications that run on a 64-bit Intel-based Mac OS* X system (also referred to as Intel® 64 architecture)

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

1.6 Relocating Product After Install

If you wish to move the installed product to a different location on disk, you can do so using a supplied script.

1. Open a terminal window
2. Change directory (cd) to the compiler install folder (for example, `/opt/intel/Compiler/11.0/xxx`)
3. Type the command:
   `./move_cproc.sh <new-install-location>`
   where `<new-install-location>` is the new directory path

This script will move all the files and update symlinks, environment variables and startup scripts as needed. If you have both Intel C++ and Intel Fortran installed in the old path, both products will be moved to the new location.

1.7 Removal/Uninstall

It is not possible to remove the compiler while leaving any of the performance library components installed.

1. Open Terminal and set default (cd) to any folder outside `<install-dir>`
2. Type the command: `<install-dir>/uninstall_cproc.sh`
3. Follow the prompts

If you are not currently logged in as `root` you will be asked for the `root` password. If you also have the same-numbered version of Intel® Fortran Compiler installed, it may also be removed.

1.8 Documentation

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

1.9 Technical Support

If you did not register your compiler during installation, please do so 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/cmac .

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

2 Intel® C++ Compiler

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

2.1 Compatibility

In version 11.0, the default for OpenMP applications is to link against the dynamic form of the "compatibility" OpenMP libraries. See below for more information.

2.2 New and Changed Features

Please refer to the compiler documentation for details

- Additional features from C++ 0x
- C++ lambda functions
- Decimal floating point
- valarray implementation using IPP option
- #pragma vector_nontemporal
- #pragma unroll_and_jam
- Support for OpenMP* 3.0

2.3 New and Changed Compiler Options

Please refer to the compiler documentation for details

- -axSSE2
- -axSSE3
- -axSSSE3

Intel® C++ Compiler Professional Edition
11.0 for Mac OS* X Installation Guide and Release Notes

- -axSSE4.1
- -axSSE4.2
- -diag-error-limit
- -diag-once
- -falign-stack
- -fasm-blocks
- -fast-transcendentals
- -ffreestanding
- -fma
- -fnon-call-exceptions
- -fp-relaxed
- -fpie
- -fstack-protector
- -help-pragma
- -m32
- -m64
- -mia32
- -minstruction
- -mssse3
- -msse4.1
- -multiple-processes
- -openmp-link
- -openmp-task
- -opt-block-factor
- -opt-calloc
- -opt-jump-tables
- -opt-subscript-in-range
- -prof-src-dir
- -prof-src-root
- -prof-src-root-cwd
- -staticlib
- -vec
- -Werror-all
- -Wformat-security
- -xHost
- -xSSE2
- -xSSE3
- -xSSE3_ATOM
- -xSSSE3
- -xSSE4.1
- -xSSE4.2

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

2.4   Other Changes

2.4.1   Environment Setup Script Changed

The icc.sh (icc.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, there is one version of these scripts and they now take an argument to select the target platform.

The command takes the form:

```
source /opt/intel/Compiler/11.0/xxx/bin/iccvars.sh argument
```

Where *xxx* is the package identifier and *argument* is either `ia32` or `intel64` as described above under [Installation Folders](). If you have installed the compiler into a different path, make the appropriate adjustments in the command. Establishing the compiler environment also establishes the Intel® Debugger (idb) environment.

2.4.2   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 gcc* compilers. These "compatibility" libraries can provide higher performance than the older "legacy" libraries. In version 11, 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.

2.4.3   OpenMP* Libraries Default to Dynamic Linking

As of version 11.0, OpenMP applications link to the dynamic OpenMP libraries by default. To specify static linking of the OpenMP libraries, specify `-openmp-link static` .

2.4.4   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.

2.5   Known Issues

2.5.1   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:

Intel® C++ Compiler Professional Edition
11.0 for Mac OS* X Installation Guide and Release Notes

```
../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 Intel® Debugger (IDB)

3.1 Known Problems

3.1.1 Dwarf vs. Stabs Debug Formats

The debugger only supports debugging of executables whose debug information is in Dwarf format, and does not support the Stabs debug format. Use the `-gdwarf-2` flag on the compile command to have gcc and g++ generate Dwarf output. The Intel compilers (icc and ifort) produce Dwarf debug format.

3.1.2 Compilation Requirements

Starting with Xcode 2.3, the Dwarf debugging information is stored in the object (.o) files. These object files are accessed by the debugger to obtain information related to the application being debugged and thus must be available for symbolic debugging.

In cases where a program is compiled and linked in one command, such as:

```
icc -g -o hello.exe hello.c
```

the object files are generated by the compiler but deleted before the command completes. The binary file produced by this command will have no debugging information. To make such an application debuggable users have two options.

Users may build the application in two steps, explicitly producing a .o file:

```
icc -c -g -o hello.o hello.c

icc -g -o hello.exe hello.o
```

Alternatively, users may use the compiler switch -save-temps to prevent the compiler from deleting the .o files it generates:

```
icc -g -save-temps -o hello.exe hello.c
```

3.1.3   Non-local Binary and Source File Access

The debugger cannot access binary files from a network-mounted file system (such as NFS). The error message will look like this:

Internal error: cannot create absolute path for: /home/me/hello

You cannot debug "/home/me/hello" because its type is "unknown".

The debugger cannot access source files from a network-mounted file system (such as NFS). The error message will look like this:

Source file not found or not readable, tried...

./hello.c

/auto/mount/site/foo/usr1/user_me/c_code/hello.c

(Cannot find source file hello.c)

The file-path specified will be correct.

The workaround in both cases is to copy the files to a local file system (i.e., one which is not mounted over the network).

3.1.4   Debugging applications that fork

Debugging the child process of an application that calls fork is not yet supported.

3.1.5   Debugging applications that exec

The $catchexecs control variable is not supported.

3.1.6   Snapshots

Snapshots are not yet supported as described in the manual.

3.1.7   Debugging optimized code

Debugging optimized code is not yet fully supported. The debugger may not be able to see some function names, parameters, variables, or the contents of the parameters and variables when code is compiled with optimizations turned on.

3.1.8   Watchpoints

Watchpoints that are created to detect write access don't trigger when a value identical to the original has been written. These restrictions are due to a limitation in the Mac OS* X operating system.

Because the SIGBUS signal rather than the SIGSEGV signal is used by the debugger to implement watchpoints, you cannot create a signal detector which will catch a SIGBUS signal.

3.1.9   Graphical User Interface (GUI)

This version of the debugger does not support the GUI

3.1.10 MPP Debugging Restrictions
MPP debugging is not supported as described in the manual.

3.1.11 Cannot debug C++ method returning class instance
When the compiler generates code for a C++ method which returns a class instance, it often generates code that allocates memory for the return value in the caller's scope and passes a pointer to that memory as a first parameter. The generated code for the called method will expect that parameter in all calls.

As an example, for user code which looks like this:

```
FooClass f = returnFooInstance();
```

...the compiler will have generated code more like this:

```
FooClass f;
FooClass temp;
returnFooInstance( &temp );
f = temp;
```

The debug information generated by the compiler describes such a method as having no return value and having that first parameter. However, despite having this information, the debugger cannot successfully call methods which have been implemented in this way. The message "Error: not enough parameters to call" will be issued by the debugger in these cases.

There is no work-around. Specifying an appropriate address as the first parameter does not work.

3.1.12 Function Breakpoints
Debugger breakpoints set in functions (using the "stop in" command) may not halt user program execution at the first statement. This is due to insufficient information regarding the function prologue in the generated Dwarf debug information. As a work-around, use the "stop at" command to set a breakpoint on the desired statement.

The compiler generates a call to "__dyld_func_lookup" as part of the prologue for some functions. If you set a breakpoint on this function the debugger will stop there, but local variable values may not be valid. The work-around is to set a breakpoint on the first statement inside the function.

3.1.13 Core File Debugging
Debugging core files is not yet supported.

3.1.14 Universal Binary Support
Debugging of universal binaries is supported. The debugger only supports debugging the IA-32 Dwarf sections of the binaries on IA-32 and either the IA-32 or the Intel® 64 sections on Intel® 64.

3.1.15 Debugger variable $threadlevel

The manual's discussion of the debugger variable "$threadlevel" says "On Mac OS* X, the debugger supports POSIX threads, also known as pthreads." This sentence might be read as implying that other kinds of threads might be supported. This is not true; only POSIX threads are supported on Mac OS* X.

4   Intel® Integrated Performance Primitives

This section summarizes changes, new features and late-breaking news about Intel® Integrated Performance Primitives (Intel® IPP).

4.1   Change History

- New function implementation in Image Processing domain ippiCopy* and ippiTranspose* functions
- Other new function implementations in speech coding and signal processing domains. Check "NewFunctionsList.txt" in the documentation directory for more details
- New unified image codec (UIC) frameworks implementation to standardize the interfaces as plug-and-play of various image codecs
- Intel® Atom™ Processor support
- High-level Data Compression library Support lzo and new continued performance improvement for zlib, gzip, bzip2
- A new sample for DMIP Deferred Mode of Image Processing over Intel IPP binary and API
- Intel® QuickAssist functional API for Cryptography
- New Domain - Data Integrity Functions based on operations over finite fields for error-correcting coding
- Generated domain/functionality (Spiral)
- Video Enhancement Denoising / Deinterlasing / Demosaicing
- Image Search descriptors (MPEG7), Color layout, Edge Histogram
- Microsoft RT Audio Support (enchancement)
- New Speech Coding Standard G729.1 Codec Support
- Super Resolution Technology, Optical Flow
- New Video AVS Codec Support for Decoding
- New Image Processing functions for 3D Support, Geom WarpAffine
- New Cryptography function support for Reed-Solomon Algorithm
- Threaded Static Libraries
- ALS Decoder Profile support in AAC Decoding

5   Intel® Math Kernel Library

This section summarizes changes, new features and late-breaking news about Intel® Math Kernel Library (Intel® MKL).

5.1 New and Changed Features

- Performance Improvements in the BLAS:
  - 32-bit improvements
    - 40-50% improvement for (Z,C)GEMM on Quad-Core Intel® Xeon® processor 5300 series
    - 10% improvement for all GEMM code on Quad-Core Intel® Xeon® processor 5400 series
  - 64-bit improvements
    - 2.5-3% improvement for DGEMM on 1 thread on Quad-Core Intel® Xeon® processor 5400 series
    - 50% improvement for SGEMM on the Intel® Core™ i7 processor family
    - 3% improvement for CGEMM on 1 thread on the f Intel® Core™ i7 processor family
    - 2-3% improvement for ZGEMM on 1 thread on the Intel® Core™ i7 processor family
    - 30% improvement for right-side cases of DTRSM on the Intel® Core™ i7 processor family
- Improvements to the direct sparse solver (DSS/PARDISO):
  - The performance of out-of-core PARDISO was improved by 35% on average.
  - Support of separate backward/forward substitution for DSS/PARDISO has been added.
  - A new parameter for turning off iterative refinement for DSS interface has been introduced.
  - A new parameter for checking sparse matrix structure has been introduced for PARDISO interface.
- The capability to track the progress of a lengthy computation and/or interrupt the computation has been added via a callback function mechanism. A function called mkl_progress can be defined in a user application, which will be called regularly from a subset of the MKL LAPACK routines. See the LAPACK Auxiliary and Utility Routines chapter in the reference manual for more information. Refer to the specific function descriptions to see which LAPACK functions support the feature.
- Transposition functions have been added to Intel MKL. See the reference manual for further detail.
- The C++ std::complex type can now be used instead of MKL-specific complex types.
- An implementation of the Boost uBLAS matrix-matrix multiplication routine is now provided which will make use of the highly optimized version of DGEMM in the Intel MKL BLAS. See the User guide for more information.
- Improvements to the sparse BLAS:
  - Support for all data types (single precision, complex and double complex) has been added.
  - Routines for computing the sum and product of two sparse matrices stored, both stored in the compressed sparse row format have been added.

- The Vector Math Library functions, CdfNorm, CdfNormInv, and ErfcInv, have been optimized to achieve much improved performance.
- Performance improvement on the Intel® Core™ i7 processor family:
  - 3-17% improvement for the following VML functions: Asin, Asinh, Acos, Acosh, Atan, Atan2, Atanh, Cbrt, CIS, Cos, Cosh, Conj, Div, ErfInv, Exp, Hypot, Inv, InvCbrt, InvSqrt, Ln, Log10, MulByConj, Sin, SinCos, Sinh, Sqrt, Tanh.
  - 7-67% improvement for uniform random number generation.
  - 3-10% improvement for VSL distribution generators based on Wichmann-Hill, Sobol, and Niederreiter BRNGs (64-bit only).
- The configuration file functionality has been removed. See the user guide for alternative means to configure the behavior of Intel MKL.
- When functions in Intel MKL are called from an MPI program they will be run on 1 thread by default (i.e., in the absence of explicit controls).
- The following VML functions: CdfNorm, CdfNormInv, and ErfcInv.
- The DftiCopyDescriptor function.
- The LP64 interface of DSS/PARDISO now uses 64-bit addressing for internal arrays on 64-bit operating systems. This allows the direct solver to solve larger systems.
- The default OpenMP runtime library for Intel MKL has been changed from libguide to libiomp. See the User Guide in the doc directory for more information.
- The interval linear solver functions have been removed from MKL.
- Support for Intel MPI 1.x has ended.
- Documentation updates:
  - The FFTW Wrappers for MKL Notes have been removed from the product package after their content was integrated into the Intel MKL Reference Manual (Appendix G).
  - New functions have been documented in the reference manual, and support for Boost uBLAS matrix-matrix multiplication has been described in the User Guide.
  - The parallel BLAS (PBLAS) which support ScaLAPACK are now documented in the Intel MKL reference manual.
  - Added FORTRAN 77 support info to the description of VML and VSL functions in the Intel MKL reference manual.

5.2 Known Limitations

5.2.1 Limitations to the sparse solver and optimization solvers:
- Sparse and optimization solver libraries functions are only provided in static form

5.2.2 Limitations to the FFT functions:
- Mode DFTI_TRANSPOSE is implemented only for the default case

- Mode DFTI_REAL_STORAGE can have the default value only and cannot be set by the DftiSetValue function (i.e. DFTI_REAL_STORAGE = DFTI_REAL_REAL)

- The ILP64 version of Intel® MKL does not currently support FFTs with any one dimension larger than 2^31-1. Any 1D FFT larger than 2^31-1 or any multi-dimensional FFT with any

Intel® C++ Compiler Professional Edition
11.0 for Mac OS* X Installation Guide and Release Notes

dimension greater than 2^31-1 will return the "DFTI_1D_LENGTH_EXCEEDS_INT32" error message. Note that this does not exclude the possibility of performing multi-dimensional FFTs with more than 2^31-1 elements; as long as any one dimension length does not exceed 2^31-1

- Some limitations exist on arrays sizes for Cluster FFT functions. See mklman.pdf for a detailed description

- When a dynamically linked application uses Cluster FFT functionality, it is required to put the static Intel® MKL interface libraries on the link line as well. For example: -Wl,--start-group $MKL_LIB_PATH/libmkl_intel_lp64.a $MKL_LIB_PATH/libmkl_cdft_core.a -Wl,--end-group $MKL_LIB_PATH/libmkl_blacs_intelmpi20_lp64.a -L$MKL_LIB_PATH -lmkl_intel_thread -lmkl_core -liomp5 -lpthread

5.2.3   Limitations to the LAPACK functions:
- The ILAENV function, which is called from the LAPACK routines to choose problem-dependent parameters for the local environment, cannot be replaced by a user's version

- second() and dsecnd() functions may not provide correct answers in the case where the CPU frequency is not constant.

5.2.4   Limitations to the Vector Math Library (VML) and Vector Statistical Library (VSL) functions:
- Usage of mkl_vml.fi may produce warning about TYPE ERROR_STRUCTURE length

5.2.5   Limitations to the ScaLAPACK functions:
- The user can not substitute PJLAENV for their own version. This function is called by ScaLAPACK routines to choose problem-dependent parameters for the local environment.

- ScaLAPACK libraries are available only in static form

5.2.6   Limitations to the ILP64 version of Intel® MKL:
- The ILP64 version of Intel® MKL does not contain the complete functionality of the library. For a full listing of what is in the ILP64 version refer to the user's guide in the doc directory.

- g77 cannot be used with the ILP64 libraries.

5.2.7   Limitations to the Fortran 95 interface to LAPACK:
- If you are compiling the Fortran 95 interface to LAPACK with the GNU gfortran compiler, you must manually remove the "pure" attribute from all subroutines containing a procedure argument: ?GEES, ?GEESX, ?GGES, ?GGESX (where ? can be S, D, C, or Z).

5.2.8   Limitations to the g77 compiler support:
- Some Intel® MKL functions contain underscore in their names (i.e. mkl_dcsrsymv, mkl_cspblas_dcsrsymv) and these functions don't support the g77 default naming convention. -fno-second-underscore compilation flag can be used as workaround for this limitation. E.g.: g77 -fno-second-underscore test.f

5.2.9 Other Limitations

- The DHPL_CALL_CBLAS option is not allowed when building the hybrid version of MP LINPACK.
- On Intel® 64 architecture processors user programs compiled with the GNU Fortran compiler (version 3.2.3) will likely get incorrect results from those functions in Intel® MKL that return single precision values, if -fno-f2c GNU Fortran compiler flag isn't used. The GNU Fortran compiler by default expects REAL*4 values in the first 8 bytes of the return register (just as a double precision value would be represented) while the Intel® Fortran compiler expects REAL*4 values in the first 4 bytes of the return register. The behavior of Intel® MKL is compatible with that of the Intel Fortran compiler. GNU Fortran compiler behavior could be changed to be compatible with the Intel Fortran compiler by using the -fno-f2c flag.
- FFT and PDE Support functions cannot be called from Fortran-77. These components have Fortran-90/95 interface specifics (structures, ..) that cannot be used with Fortran-77.
- We recommend that -Od be used when compiling test source code available with Intel® MKL. Current build scripts do not specify this option and default behavior for the compilers has changed to provide vectorization.
- All VSL functions return an error status, i.e., default VSL API is a function style now rather than a subroutine style used in earlier Intel® MKL versions. This means that Fortran users should call VSL routines as functions. For example:
  errstatus = vslrnggaussian(method, stream, n, r, a, sigma)
  rather than subroutines:
  call vslrnggaussian(method, stream, n, r, a, sigma)
  Nevertheless, Intel® MKL provides a subroutine-style interface for backward compatibility. To use subroutine-style interface, manually include mkl_vsl_subroutine.fi file instead of mkl_vsl.fi by changing the line include 'mkl_vsl.fi' mkl.fi (in the include directory) with the line include 'mkl_vsl_subroutine.fi'. VSL API changes don't affect C/C++ users.

5.3 Memory Allocation

In order to achieve better performance, memory allocated by Intel® MKL is not released. This behavior is by design and is a onetime occurrence for Intel® MKL routines that require workspace memory buffers. Even so, the user should be aware that some tools may report this as a memory leak. Should the user wish, memory can be released by the user program through use of a function (MKL_FreeBuffers()) made available in Intel® MKL or memory can be released after each call by setting the environment variable MKL_DISABLE_FAST_MM (see User's Guide in the doc directory for more details). Using one of these methods to release memory will not necessarily stop programs from reporting memory leaks, and in fact may increase the number of such reports should you make multiple calls to the library thereby requiring new allocations with each call. Memory not released by one of the methods described will be released by the system when the program ends. To avoid this restriction disable memory management as described above.

5.4   Other Notes

The GMP component is located in the solver library. For Intel® 64 platforms these components support only LP64 interface.

6   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.

7   Disclaimer and Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL(R) 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.

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.

Copyright © 2009 Intel Corporation. All Rights Reserved.