Table of Contents

1 Introduction.................................................................................................................................................. 3
  1.1 Change History ....................................................................................................................................... 3
  1.2 Product Contents .................................................................................................................................... 3
  1.3 System Requirements ............................................................................................................................. 3
    1.3.1 IA-64 Architecture (Intel® Itanium®) Development Not Supported ........................................... 5
  1.4 Documentation ....................................................................................................................................... 5
  1.5 Samples .................................................................................................................................................. 6
  1.6 Japanese Language Support .................................................................................................................. 6
  1.7 Technical Support ................................................................................................................................... 6

2 Installation .................................................................................................................................................. 7
  2.1 Pre-Installation Steps ............................................................................................................................ 7
    2.1.1 Configure Visual Studio for 64-bit Applications ........................................................................... 7
    2.1.2 Installation on Microsoft Windows Vista® and Windows 7® ....................................................... 7
  2.2 Installation ................................................................................................................................................ 7
    2.2.1 Activation of Purchase after Evaluation Using the Intel Activation Tool ...................................... 8
    2.2.2 Using a License Server ................................................................................................................... 8
    2.2.3 Installing the IMSL® Fortran Numerical Library® ....................................................................... 8
    2.2.4 Prompt for Administrator Permission with Microsoft Visual Studio 2005® ............................ 8
  2.3 Changing, Updating and Removing the Product .................................................................................... 9
  2.4 Installation Folders ................................................................................................................................ 9
  2.5 Installation Known Issues ....................................................................................................................... 10
    2.5.1 Additional Steps to Install Documentation for Microsoft Visual Studio 2010 ....................... 10
    2.5.2 Documentation Issue with Multiple Visual Studio Versions .................................................... 11

3 Intel® Visual Fortran Compiler .................................................................................................................. 11
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Compatibility</td>
<td>11</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Stack Alignment Change for REAL(16) and COMPLEX(16) Datatypes</td>
<td>12</td>
</tr>
<tr>
<td>3.2</td>
<td>New and Changed Compiler Features</td>
<td>12</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Features from Fortran 2003</td>
<td>12</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Features from Fortran 2008</td>
<td>12</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Coarrays</td>
<td>13</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Other Changes</td>
<td>14</td>
</tr>
<tr>
<td>3.3</td>
<td>New and Changed Compiler Options</td>
<td>14</td>
</tr>
<tr>
<td>3.4</td>
<td>Other Changes</td>
<td>15</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Build Environment Command Script Change</td>
<td>15</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Static Security Analysis Feature (formerly Source Checker) Requires Intel® Inspector XE</td>
<td>16</td>
</tr>
<tr>
<td>3.4.3</td>
<td>OpenMP* Legacy Libraries Removed</td>
<td>16</td>
</tr>
<tr>
<td>3.4.4</td>
<td>OpenMP* Libraries Default to Dynamic Linking</td>
<td>16</td>
</tr>
<tr>
<td>3.4.5</td>
<td>IMSL Licensing Changes for Redistribution</td>
<td>16</td>
</tr>
<tr>
<td>3.4.6</td>
<td>RANF Portability Function Is Now an Intrinsic</td>
<td>16</td>
</tr>
<tr>
<td>3.5</td>
<td>Known Issues</td>
<td>17</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Command-Line Diagnostic Issue for Filenames with Japanese Characters</td>
<td>17</td>
</tr>
<tr>
<td>3.6</td>
<td>Microsoft Visual Studio 2010* Notes</td>
<td>17</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Configuring Microsoft Visual C++ to Reference Intel® Fortran Run-Time Libraries</td>
<td>17</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Adjusting Project Dependencies</td>
<td>18</td>
</tr>
<tr>
<td>3.7</td>
<td>Fortran 2003 and Fortran 2008 Feature Summary</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>Intel® Math Kernel Library</td>
<td>22</td>
</tr>
<tr>
<td>4.1</td>
<td>What's New in Intel® MKL 10.3 Update 1</td>
<td>22</td>
</tr>
<tr>
<td>4.2</td>
<td>What's New in Intel® MKL 10.3</td>
<td>22</td>
</tr>
<tr>
<td>4.3</td>
<td>Known Issues</td>
<td>24</td>
</tr>
<tr>
<td>4.4</td>
<td>Notices</td>
<td>24</td>
</tr>
<tr>
<td>4.5</td>
<td>Attributions</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>Intel® Parallel Debugger Extension</td>
<td>25</td>
</tr>
<tr>
<td>5.1</td>
<td>New Features</td>
<td>25</td>
</tr>
<tr>
<td>5.2</td>
<td>Known Issues</td>
<td>25</td>
</tr>
<tr>
<td>5.3</td>
<td>Documentation</td>
<td>26</td>
</tr>
<tr>
<td>6</td>
<td>Disclaimer and Legal Information</td>
<td>27</td>
</tr>
</tbody>
</table>
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.


1.1 Change History
This section highlights important changes in product updates.

Update 1 (12.0.1)
- Intel® Math Kernel Library updated to 10.3 Update 1
- Corrections to reported problems

Product Release (12.0.0)
- Initial product release

1.2 Product Contents
Intel® Visual Fortran Composer XE 2011 for Windows* includes the following components:

- Intel® Visual Fortran Compiler XE 12.0.1 for building applications that run on IA-32 and Intel® 64 architecture systems
- Intel® Math Kernel Library 10.3 Update 1
- Integration into Microsoft* development environments
- Intel® Parallel Debugger Extension for Microsoft Visual Studio*
- Microsoft Visual Studio 2008 Shell and Libraries (not included with Student or Evaluation licenses)
- Sample programs
- On-disk documentation

Intel® Visual Fortran Composer XE 2011 with IMSL* for Windows* includes the above plus the IMSL* Fortran Numerical Library* from Visual Numerics*

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

- 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
  - For the best experience, a multi-core or multi-processor system is recommended
• 1GB RAM (2GB recommended)
• 2GB free disk space required for all product features and all architectures
• To use the Microsoft Visual Studio development environment or command-line tools to build IA-32 or Intel® 64 architecture applications, one of:
  o Microsoft Visual Studio 2010* with C++ and “X64 Compiler and Tools” components installed [1]
  o Microsoft Visual Studio 2008* Standard Edition or higher with C++ and “X64 Compiler and Tools” components installed [1]
  o Microsoft Visual Studio 2005* Standard Edition or higher with C++ and “X64 Compiler and Tools” components installed [1]
  o Intel® Visual Fortran development environment based on Microsoft Visual Studio 2008 Shell (included with some license types of Intel® Fortran Compiler) [2]
• To use command-line tools only to build IA-32 architecture applications, one of:
• To use command-line tools only to build Intel® 64 architecture applications, one of:
  o Microsoft Windows Software Development Kit Update for Windows Vista*
  o Microsoft Windows SDK for Windows 2008 and .NET Framework 3.5*
• To read the on-disk documentation, Adobe Reader* 7.0 or later

Notes:

1. Microsoft Visual Studio 2005 and 2008 Standard Edition installs the “x64 Compiler and Tools” component by default – the Professional and higher editions require a “Custom” install to select this. Microsoft Visual Studio 2010 installs this component by default in all editions.
2. Intel® Visual Fortran development environment based on Microsoft Visual Studio 2008 Shell is included with Academic and Commercial licenses for Intel Visual Fortran Composer XE. It is not included with Evaluation or Student licenses. This development environment provides everything necessary to edit, build and debug Fortran applications. Some features of the full Visual Studio product are not included, such as:
   • Resource Editor (see ResEdit* (http://www.resedit.net/), a third-party tool, for a substitute)
   • Automated conversion of Compaq* Visual Fortran projects
   • Microsoft language tools such as Visual C++ or Visual Basic*
3. If you will be installing Microsoft Visual Studio 2008 Shell and you wish to also use Microsoft Visual C++ 2008 Express Edition (for separate access to the Microsoft C++ compiler), you must uninstall Visual C++ 2008 Express Edition before installing Visual Studio 2008 Shell along with Intel Visual Fortran Compiler. After the Fortran install is complete, you may reinstall Visual C++ 2008 Express Edition if desired. Please note that the Fortran and C++ compiler environments will be separate and not combined.

4. The default for Intel® Visual Fortran is to build IA-32 architecture applications that require a processor supporting the Intel® SSE2 instructions. A compiler option is available to generate code that will run on any IA-32 architecture processor.

5. Applications can be run on the same Windows versions as specified above for development. Applications may also run on non-embedded 32-bit versions of Microsoft Windows earlier than Windows XP, though Intel does not test these for compatibility. Your application may depend on a Win32 API routine not present in older versions of Windows. You are responsible for testing application compatibility. You may need to copy certain run-time DLLs onto the target system to run your application.

1.3.1 IA-64 Architecture (Intel® Itanium®) Development Not Supported
This product version does not support development on or for IA-64 architecture (Intel® Itanium®) systems. The version 11.1 compiler remains available for development of IA-64 architecture applications.

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

**Optimization Notice**

Intel® compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel® and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the “Intel® Compiler User and Reference Guides” under “Compiler Options.” Many library routines that are part of Intel® compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel® compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel® compilers, associated libraries and associated development tools may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3) instruction sets and other optimizations. Intel does not
guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel® and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20101101

### 1.5 Samples

Samples for each product component can be found in the `Samples` folder as shown under `Installation Folders`.

### 1.6 Japanese Language Support

Intel compilers provide support for Japanese language users when the combined Japanese-English installation is used. 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


### 1.7 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/

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

2.1 Pre-Installation Steps

2.1.1 Configure Visual Studio for 64-bit Applications
If you are using Microsoft Visual Studio 2005* or 2008 and will be developing 64-bit applications (for the Intel® 64 architecture) you may need to change the configuration of Visual Studio to add 64-bit support.

If you are using Visual Studio 2005/2008 Standard Edition or Visual Studio 2008 Shell, no configuration is needed to build Intel® 64 architecture applications. For other editions:

2. Click Add or Remove Features
3. Under “Select features to install”, expand Language Tools > Visual C++
4. If the box “X64 Compiler and Tools” is not checked, check it, then click Update. If the box is already checked, click Cancel.

This step is not required when using Microsoft Visual Studio 2010.

2.1.2 Installation on Microsoft Windows Vista* and Windows 7*
Microsoft Visual Studio 2005* users should install Visual Studio 2005 Service Pack 1 (VS 2005 SP1) as well as the Visual Studio 2005 Service Pack 1 Update for Windows Vista, which is linked to from the VS 2005 SP1 page. After installing these updates, you must ensure that Visual Studio runs with Administrator permissions, otherwise you will be unable to use the Intel compiler. For more information, please see Microsoft's Visual Studio on Windows Vista page (http://msdn2.microsoft.com/en-us/vstudio/aa948853.aspx) and related documents. See also Prompt for Administrator Permission with Microsoft Visual Studio 2005*. 

2.2 Installation
The installation of the product requires a valid license file or serial number. If you are evaluating the product, you can also choose the “Evaluate this product (no serial number required)” option during installation.

If you received your product on DVD, insert the first product DVD in your computer’s DVD drive; the installation should start automatically. If it does not, open the top-level folder of the DVD drive in Windows Explorer and double-click on setup.exe.

If you received your product as a downloadable file, double-click on the executable file ( .EXE) to begin installation. Note that there are several different downloadable files available, each providing different combinations of components. Please read the download web page carefully to determine which file is appropriate for you.
You do not need to uninstall previous versions or updates before installing a newer version – the new version will coexist with the older versions. If you want to remove older versions, you may do so before or after installing the newer one.

2.2.1 Activation of Purchase after Evaluation Using the Intel Activation Tool
Note for evaluation customers: a new tool Intel Activation Tool “ActivationTool.exe” is included in this product release and installed at “[Common Files]\Intel\Parallel Studio XE 2011\Activation\”.

If you installed the product using an Evaluation license or serial number (SN), or using the “Evaluate this product (no serial number required)” option during installation, and then purchased the product, you can activate your purchase using the Intel Activation Tool at Start > All Programs > Intel Parallel Studio XE 2011 > Product Activation. It will convert your evaluation software to a fully licensed product.

2.2.2 Using a License Server
If you have purchased a “floating” license, see http://software.intel.com/en-us/articles/licensing-setting-up-the-client-floating-license/ for information on how to install using a license file or license server. This article also provides a source for the Intel® License Server that can be installed on any of a wide variety of systems.

2.2.3 Installing the IMSL® Fortran Numerical Library®
If you have Intel Visual Fortran Composer XE with IMSL®, the IMSL installation is separate from the compiler installation: either a separate download or a separate disc. You must install the compiler before installing IMSL.

2.2.4 Prompt for Administrator Permission with Microsoft Visual Studio 2005®
If you are installing on Microsoft Windows Vista® or later versions of Microsoft Windows and are using Microsoft Visual Studio 2005, Windows may display a dialog similar to the following:
If this is displayed, it is important that you click the Continue button and leave the “Always show this message” box checked. If you select “Exit Visual Studio” instead, or do nothing (this message times out after two minutes), the compiler integration will not install completely.

For more information, see Installation on Microsoft Windows Vista®.

2.3 Changing, Updating and Removing the Product
Use the Windows Control Panel “Add or Remove Products” applet to change which product components are installed or to remove the product. Depending on which product you installed, the entry will be one of the following:

- Intel(R) Visual Fortran Composer XE 2011 for Windows®
- Intel(R) Composer XE 2011 for Windows®
- Intel(R) Parallel Studio XE 2011 for Windows®

If you also installed Microsoft Visual Studio 2008 Shell as part of the compiler install, the following additional entries will be present:

- Microsoft Visual Studio 2008 Shell ENU
- Microsoft Tools and Libraries for Intel(R) Visual Fortran

These entries should not be removed unless you want to completely remove the product.

When installing an updated version of the product, you do not need to remove the older version first. The first time you install an update, you will have the choice to replace the older version or to keep both the older and newer versions on the system. This choice is remembered for future updates. In Microsoft Visual Studio you can select which specific compiler version to use through the Tools > Options > Intel(R) Visual Fortran Compiler dialog. Compiler versions older than 11.0 are not available to be selected through Visual Studio. All installed versions can be used from the command line.

If you remove a newer version of the product you may have to reinstall the integrations into Microsoft Visual Studio from the older version.

2.4 Installation Folders
The installation folder arrangement is shown in the diagram below. Not all folders will be present in a given installation.

- C:\Program Files\Intel\ComposerXE-2011
  o bin
    - ia32
    - ia32_intel64
    - intel64
  o compiler
    - include
      - ia32
Where 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
- **ia32_intel64**: Compilers that run on IA-32 to build applications that run on Intel® 64

If you are installing on a system with a non-English language version of Windows, the name of the `Program Files` folder may be different. On Intel® 64 architecture systems, the folder name is `Program Files (X86)` or the equivalent.

By default, updates of a given version will replace the existing directory contents. When the first update is installed, the user is given the option of having the new update installed alongside the previous installation, keeping both on the system. If this is done, the top-level folder name for the older update is changed to `ComposerXE-2011.nnn` where `nnn` is the update number.

### 2.5 Installation Known Issues

#### 2.5.1 Additional Steps to Install Documentation for Microsoft Visual Studio 2010

When installing Intel Visual Fortran Composer XE 2011 on a system with Microsoft Visual Studio 2010 for the first time, you will be asked to initialize the “Local Store” for documentation for Visual Studio 2010 if it was not done before. The “Help Library Manager” will register the Intel Visual Fortran Composer XE 2011 help documentation within Visual Studio 2010. Please follow the instructions of the "Help Library Manager" installation wizard to install the Intel Visual Fortran Composer XE 2011 help documentation for Visual Studio 2010.
This step is only needed once. When you install Intel Visual Fortran Composer XE 2011 updates in the future, you will not be required to re-register the documentation through the “Help Library Manager”.

For the more information, see http://msdn.microsoft.com/en-us/library/dd264831.aspx or search for “Help Library Manager”.

2.5.2 Documentation Issue with Multiple Visual Studio Versions
If you have both Microsoft Visual Studio* 2005 and 2008 installed on your system and integrate Intel® Visual Fortran Composer XE 2011 into both versions, removing the integration from one of the versions will remove the integrated Intel® Visual Fortran Composer XE 2011 documentation from both.

To re-install the documentation:

1. Use the Control Panel to select the product.
   - For Windows XP* users: Select Control Panel > Add/Remove Programs.
   - For Windows 7* users: Select Control Panel > Programs and Features.
   - For Windows Vista* users: Select Control Panel > Programs.
2. With the product selected, click the Change/Remove button and choose Modify mode.
3. In the Select Components dialog box, unselect “Integrated Documentation;” this will remove the documentation.
4. Repeat steps 1 and 2.
5. In the Select Components dialog box, select “Integrated Documentation” to install documentation again

3 Intel® Visual Fortran Compiler
This section summarizes changes, new features and late-breaking news about the Intel® Visual Fortran Compiler.

3.1 Compatibility
In general, object code and modules compiled with earlier versions of Intel Fortran Compiler (8.0 and later) may be used in a build with version 12.0. Exceptions include:

- Sources that use the CLASS keyword to declare polymorphic variables must be recompiled.
- Objects built with the multi-file interprocedural optimization (/Qipo) option must be recompiled.
- Objects that use the REAL(16) or REAL*16 datatypes must be recompiled.
- Objects built for the Intel® 64 architecture with a compiler version earlier than 10.0 and that have module variables must be recompiled. If non-Fortran sources reference these variables, the external names may need to be changed to remove an incorrect leading underscore.
• Modules that specified an ATTRIBUTES ALIGN directive and were compiled with versions earlier than 11.0 must be recompiled. The compiler will notify you if this issue is encountered.

3.1.1 Stack Alignment Change for REAL(16) and COMPLEX(16) Datatypes
In previous releases, when a REAL(16) or COMPLEX(16) (REAL*16 or COMPLEX*32) item was passed by value, the stack address was aligned at 4 bytes. For improved performance, the version 12 compiler aligns such items at 16 bytes and expects received arguments to be aligned on 16-byte boundaries.

This change primarily affects compiler-generated calls to library routines that do computations on REAL(16) values, including intrinsics. If you have code compiled with earlier versions and link it with the version 12 libraries, or have an application linked to the shared version of the Intel run-time libraries, it may give incorrect results.

In order to avoid errors, you must recompile all Fortran sources that use the REAL(16) and COMPLEX(16) datatypes.

3.2 New and Changed Compiler Features

3.2.1 Features from Fortran 2003
• FINAL subroutines
• GENERIC keyword for type-bound procedures
• A generic interface may have the same name as a derived type
• Bounds specification and bounds remapping list on a pointer assignment

3.2.2 Features from Fortran 2008
• Maximum array rank has been raised to 31 dimensions (Fortran 2008 specifies 15)
• Coarrays
  o CODIMENSION attribute
  o SYNC ALL statement
  o SYNC IMAGES statement
  o SYNC MEMORY statement
  o CRITICAL and END CRITICAL statements
  o LOCK and UNLOCK statements
  o ERROR STOP statement
  o ALLOCATE and DEALLOCATE may specify coarrays
  o Intrinsic procedures IMAGE_INDEX, LCOBOUND, NUM_IMAGES, THIS_IMAGE, UCOBOUND
  o Note: ATOMIC_DEFINE and ATOMIC_REF are not supported in this version
• CONTIGUOUS attribute
• MOLD keyword in ALLOCATE
• DO CONCURRENT
• NEWUNIT keyword in OPEN
• G0 and G0.d format edit descriptor
• Unlimited format item repeat count specifier
• A CONTAINS section may be empty
• Intrinsic procedures BESSEL_J0, BESSEL_J1, BESSEL_JN, BESSEL_YN, BGE, BGT, BLE, BLT, DSHIFTL, DSHIFTR, ERF, ERFC, ERFC_SCALED, GAMMA, HYPOT, IALL, IANY, IPARITY, IS_CONTIGUOUS, LEADZ, LOG_GAMMA, MASKL, MASKR, MERGE_BITS, NORM2, PARITY, POPCNT, POPPAR, SHIFTA, SHIFTL, SHIFTR, STORAGE_SIZE, TRAILZ,
• Additions to intrinsic module ISO_FORTRAN_ENV: ATOMIC_INT_KIND, ATOMIC_LOGICAL_KIND, CHARACTER_KINDS, INTEGER_KINDS, INT8, INT16, INT32, INT64, LOCK_TYPE, LOGICAL_KINDS, REAL_KINDS, REAL32, REAL64, REAL128, STAT_LOCKED, STAT_LOCKED_OTHER_IMAGE, STAT_UNLOCKED

3.2.3 Coarrays
No special procedure is necessary to run a program that uses coarrays; you simply run the executable file. The underlying parallelization implementation is Intel® MPI. Installation of the compiler automatically installs the necessary Intel® MPI run-time libraries to run on shared memory. The Intel® Cluster Toolkit installs the necessary Intel® MPI run-time libraries to run on distributed memory. Use of coarray applications with any other MPI implementation, or with OpenMP®, is not supported.

By default, the number of images created is equal to the number of execution units on the current system. You can override that by specifying the option /Qcoarray-num-images:<n> on the ifort command that compiles the main program. You can also specify the number of images in an environment variable FOR_COARRAY_NUM_IMAGES.

3.2.3.1 Specifying Shared or Distributed Memory Processing of Coarrays
The documentation for the /Qcoarray option currently says:

Using /Qcoarray (Windows*) or -coarray (Linux*) with no argument is equivalent to running on multi-node (distributed memory) if an Intel® Cluster Toolkit license is installed or on single node (shared memory) if there is no Intel® Cluster Toolkit license installed.

The implementation has changed since the above text was written. The new behavior is that if /Qcoarray is specified without the memory argument, shared memory is used whether or not the Intel® Cluster Toolkit license is present. To use distributed memory, which requires that a license for Intel® Cluster Toolkit is present, specify /Qcoarray:distributed.

3.2.3.1 Coarray Known Issues
The following features are known not to work in this version:

• Coarrays are not supported on a distributed memory environment.
• Character data type coarrays
• Coarrays of derived type where the type contains an ultimate component that is ALLOCATABLE or POINTER
• Output (WRITE, PRINT, etc.) of an array slice of a coarray referencing another image. A whole array reference, or a single element works.
• Default initialization of a REAL(16) or COMPLEX(16) coarray
• LOCK and UNLOCK cannot be used on another image.
• STAT= or ERRMSG= arguments on LOCK, UNLOCK, SYNC IMAGES, SYNC MEMORY, or SYNC ALL are not being set correctly.

3.2.4 Other Changes
• The ability to create a source listing file with identifier cross-reference has been added
• Guided auto-parallelism
• An option to use math library functions that are faster but return results with less precision or accuracy
• An option to use math library functions that return consistent results across different models and manufacturers of processors
• An option to simplify specifying that Fortran 2003 semantics should be assumed for all syntax where the compiler default is not consistent with Fortran 2003
• The ability to generate a build dependencies output file has been added
• In Visual Studio projects, the module output directory of a subproject is automatically added as an “additional INCLUDE directory” for its parent project

3.3 New and Changed Compiler Options
Please refer to the compiler documentation for details

• /assume:[no]fpe_summary
• /assume:old_ldout_format
• /gen-dep
• /gen-depformat
• /list
• /list-line-len
• /list-page-len
• /Qcoarray (see above)
• /Qcoarray-num-images
• /Qcov-dir
• /Qcov-file
• /Qcov-gen
• /Qdiag-sc-dir
• /Qfp-trap
• /Qfp-trap-all
• /Qguide
• /Qguide-data-trans
• /Qguide-file
• /Qguide-file-append
• /Qguide-opts
• /Qguide-par
The command window script used to establish the build environment allows the optional specification of the version of Microsoft Visual Studio to use. If you are not using the predefined Start menu shortcut to open a build environment window, use the following command to establish the proper environment:

"<install-dir>\bin\compilervars.bat" arch [vs]

Where arch is one of ia32, ia32_intel64 or intel64 as appropriate for the target architecture you want to build for. vs is optional and can be one of vs2010, vs2008 or vs2005. If vs is not specified, the version of Visual Studio specified at installation time for command-line integration is used by default.

Note: If the version of Visual Studio installed is Visual Studio 2008 Shell, you can specify vs as vs2008shell or omit it.

If you also have Intel® C++ Composer XE 2011 installed, this command will also establish the environment for using that compiler.
The script file names iclvars.bat and ifortvars.bat have been retained for compatibility with previous releases.

3.4.2 Static Security Analysis Feature (formerly Source Checker) Requires Intel® Inspector XE
The “Source Checker” feature, from compiler version 11.1, has been enhanced and renamed “Static Security Analysis”. The compiler options to enable Static Security Analysis remain the same as in compiler version 11.1 (for example, /Qdiag-enable:sc), but the results are now written to a file that is interpreted by Intel® Inspector XE rather than being included in compiler diagnostics output.

3.4.3 OpenMP® Legacy Libraries Removed
The OpenMP “legacy” libraries have been removed in this release. Only the “compatibility” libraries are provided.

3.4.4 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 /Qopenmp-link:static. The static libraries are deprecated and may be removed in a future major release.

3.4.5 IMSL Licensing Changes for Redistribution
Beginning with the release of Intel® Visual Fortran Composer XE 2011 for Windows® with IMSL® product, licensed customers who want to distribute applications built with IMSL must purchase a runtime license for IMSL. The IMSL runtime license will be available from Intel to customers who want to distribute within their own company or entity their application built with IMSL. Customers will need to contact Visual Numerics, Inc. directly if they want to distribute to third parties their applications built with IMSL. Information on purchase of the IMSL runtime license can be found at the link below.


The IMSL libraries are subject to the terms and conditions of the separate Visual Numerics, Inc’s End User License Agreement, and are not included in the compiler redistributable library package. The customer who develops the application that uses IMSL must identify the redistributable files needed for their application, and include those files separately with their application installation package.

3.4.6 RANF Portability Function Is Now an Intrinsic
The RANF function in the portability library is a non-standard random number generator. As of the version 12.0 compiler, RANF is an intrinsic function with a new, higher-performance implementation. If your program has added USE IFPORT to provide access to RANF, no changes will be seen and you will get the older version. If your program does not have USE IFPORT, or you add INTRINSIC RANF, you will get the new version that returns a different sequence, for a given seed, than the older version. The portability subroutine SRAND is still
used to set the seed for RANF. Intel recommends use of the standard intrinsic RANDOM_NUMBER, but RANF is provided for compatibility with applications already using it.

3.5 Known Issues

3.5.1 Command-Line Diagnostic Issue for Filenames with Japanese Characters
The filename in compiler diagnostics for filenames containing Japanese characters may be displayed incorrectly when compiled within a Windows command shell using the native Intel® 64 architecture compiler. It is not a problem when using Visual Studio or when using the Intel® 64 architecture cross-compiler or IA-32 architecture compiler.

3.6 Microsoft Visual Studio 2010* Notes
Microsoft Visual Studio 2010 brings several changes that primarily affect building of mixed-language applications where the main program is in C or C++.

3.6.1 Configuring Microsoft Visual C++ to Reference Intel® Fortran Run-Time Libraries
In previous releases, one used the Tools > Options > Projects and Solutions > VC++ Directories dialog to make the Intel Fortran LIB folder available to C/C++ projects. In Visual Studio 2010, the method of doing this is very different.

1. In Visual Studio, with a solution open that contains a C++ project, select View > Property Manager. If you do not see Property Manager under the View menu, you will find it under View > Additional Windows. The Property Manager window will appear. Note that this is not Properties Window or Properties Pages.
2. Click on the triangles or + signs to expand the property tree under the Debug|Win32 configuration
3. Double click on Microsoft.Cpp.Win32.user
4. Select VC++ Directories
5. Click in the field to the right of "Library Directories"
6. Click the triangle that appears to the right and select <Edit...>
7. Click the New Line button or press Ctrl-Insert
8. In the new field that appears, type:

    $(IFORT_COMPILER12)\compiler\lib\ia32

9. Click OK, OK
10. In the Visual Studio toolbar, select File > Save All

If you will be building Intel® 64 (x64) configurations:

1. Back in the Property Manager, expand the Debug|x64 configuration
2. Double click on Microsoft.Cpp.x64.user
3. Select VC++ Directories
4. Click in the field to the right of "Library Directories"
5. Click the triangle that appears to the right and select <Edit...>
6. Click the New Line button or press Ctrl-Insert
7. In the new field that appears, type:

    $(IFORT_COMPILER12)\compiler\lib\intel64

8. Click OK, OK

9. In the Visual Studio toolbar, select File > Save All

Click on the Solution Explorer tab, or press Ctrl-Alt-L, to make it visible again.

If you do not see the Microsoft.Cpp.x64.user property page listed for the x64 configuration, right click on Debug|x64 and select Add Existing property Sheet. Browse to the location which contains the MsBuild 4.0 property pages. On Windows XP, this is typically:

C:|Documents and Settings|<username>|\Local Settings\Application Data\Microsoft\MSBuild\v4.0

On Windows Vista and Windows 7, it is typically:

C:|Users|<username>|\Local Settings\AppData\Local\Microsoft\MSBuild\v4.0

You may need to enable viewing of hidden files and folders to see these paths.

Select Microsoft.Cpp.x64.user.props and click Open. Now follow the steps above.

3.6.2 Adjusting Project Dependencies

If you are converting a project from an earlier version of Visual Studio and had established Project Dependencies, these are converted to References by Visual Studio 2010. A Fortran project that is referenced by a C/C++ project will prevent the C/C++ project from building, with an MSB4075 error. To solve this:

1. Right click on the C/C++ project and select References.
2. If any Fortran project is shown as a reference, click Remove Reference. Repeat this for all Fortran projects shown as a reference. Click OK.
3. Repeat the above steps for any other C/C++ project.

Now you have to reestablish project dependencies.

1. Right click on the C/C++ project and select Project Dependencies.
2. Check the box for each project that is a dependent of this project.
3. Click OK.
4. Repeat the above steps for any other C/C++ project that has dependencies.

Unlike earlier versions of Visual Studio, Visual Studio 2010 does not automatically link in the output library of dependent projects, so you will need to add those libraries explicitly to the parent project under Linker > Additional Dependencies. You can use the Visual Studio macros $(ConfigurationName) and $(PlatformName) as required to qualify the path. For example:

..\FLIB\$(ConfigurationName)\FLIB.lib
Where \$(ConfigurationName) will expand to Release or Debug, as appropriate. Similarly, \$(PlatformName) will expand to Win32 or x64 as appropriate.

3.7 Fortran 2003 and Fortran 2008 Feature Summary

The Intel Fortran Compiler supports many features that are new in Fortran 2003. Additional Fortran 2003 features will appear in future versions. Fortran 2003 features supported by the current compiler include:

- The Fortran character set has been extended to contain the 8-bit ASCII characters `~ \[ ] ^ { } | # @`
- Names of length up to 63 characters
- Statements of up to 256 lines
- Square brackets [ ] are permitted to delimit array constructors instead of (/ /)
- Structure constructors with component names and default initialization
- Array constructors with type and character length specifications
- A named PARAMETER constant may be part of a complex constant
- Enumerators
- Allocatable components of derived types
- Allocatable scalar variables
- Deferred-length character entities
- PUBLIC types with PRIVATE components and PRIVATE types with PUBLIC components
- ERMMSG keyword for ALLOCATE and DEALLOCATE
- SOURCE= keyword for ALLOCATE
- Type extension
- CLASS declaration
- Polymorphic entities
- Inheritance association
- Deferred bindings and abstract types
- Type-bound procedures
- TYPE CONTAINS declaration
- ABSTRACT attribute
- DEFERRED attribute
- NON_OVERRIDABLE attribute
- GENERIC keyword for type-bound procedures
- FINAL subroutines
- ASYNCHRONOUS attribute and statement
- BIND(C) attribute and statement
- PROTECTED attribute and statement
- VALUE attribute and statement
- VOLATILE attribute and statement
- INTENT attribute for pointer objects
• Reallocation of allocatable variables on the left hand side of an assignment statement when the right hand side differs in shape or length (requires option /assume:realloc_lhs if not deferred-length character)
• Bounds specification and bounds remapping on a pointer assignment
• ASSOCIATE construct
• SELECT TYPE construct
• In all I/O statements, the following numeric values can be of any kind: UNIT=, IOSTAT=
• NAMELIST I/O is permitted on an internal file
• Restrictions on entities in a NAMELIST group are relaxed
• Changes to how IEEE Infinity and NaN are represented in formatted input and output
• FLUSH statement
• WAIT statement
• ACCESS='STREAM' keyword for OPEN
• ASYNCHRONOUS keyword for OPEN and data transfer statements
• ID keyword for INQUIRE and data transfer statements
• POS keyword for data transfer statements
• PENDING keyword for INQUIRE
• The following OPEN numeric values can be of any kind: RECL=
• The following READ and WRITE numeric values can be of any kind: REC=, SIZE=
• The following INQUIRE numeric values can be of any kind: NEXTREC=, NUMBER=, RECL=, SIZE=
• Recursive I/O is allowed in the case where the new I/O being started is internal I/O that does not modify any internal file other than its own
• IEEE Infinities and NaNs are displayed by formatted output as specified by Fortran 2003
• BLANK, DECIMAL, DELIM, ENCODING, IOMSG, PAD, ROUND, SIGN, SIZE I/O keywords
• DC, DP, RD, RC, RN, RP, RU, RZ format edit descriptors
• In an I/O format, the comma after a P edit descriptor is optional when followed by a repeat specifier
• Rename of user-defined operators in USE
• INTRINSIC and NON_INTRINSIC keywords in USE
• IMPORT statement
• Allocatable dummy arguments
• Allocatable function results
• PROCEDURE declaration
• Procedure pointers
• ABSTRACT INTERFACE
• PASS and NOPASS attributes
• The COUNT_RATE argument to the SYSTEM_CLOCK intrinsic may be a REAL of any kind
• Execution of a STOP statement displays a warning if an IEEE floating point exception is signaling
• MAXLOC or MINLOC of a zero-sized array returns zero if the option 
  /assume:noold_maxminloc is specified.
• Type inquiry intrinsic functions
  • COMMAND_ARGUMENT_COUNT intrinsic
  • EXTENDS_TYPE_OF and SAME_TYPE_AS intrinsic functions
  • GET_COMMAND intrinsic
  • GET_COMMAND_ARGUMENT intrinsic
  • GET_ENVIRONMENT_VARIABLE intrinsic
  • IS_IOSTAT_END intrinsic
  • IS_IOSTAT_EOR intrinsic
  • MAX/MIN/MAXVAL/MINVAL/MAXLOC/MINLOC intrinsics allow CHARACTER 
    arguments
  • MOVE_ALLOC intrinsic
  • NEW_LINE intrinsic
  • SELECTED_CHAR_KIND intrinsic
  • The following intrinsics take an optional KIND= argument: ACHAR, COUNT, IACHAR,
    ICHAR, INDEX, LBOUND, LEN, LEN_TRIM, MAXLOC, MINLOC, SCAN, SHAPE, SIZE,
    UBOUND, VERIFY
  • ISO_C_BINDING intrinsic module
  • IEEE_EXCEPTIONS, IEEE_ARITHMETIC and IEEE_FEATURES intrinsic modules
  • ISO_FORTRAN_ENV intrinsic module

Fortran 2003 features not yet supported include:

• User-defined derived type I/O
• Parameterized derived types
• A polymorphic SOURCE= specification in ALLOCATE

The Intel® Fortran Compiler also supports some features from the Fortran 2008 standard. 
Additional features will be supported in future releases. Fortran 2008 features supported by the 
current version include:

• Maximum array rank has been raised to 31 dimensions (Fortran 2008 specifies 15)
• Coarrays
  • CODIMENSION attribute
  • SYNC ALL statement
  • SYNC IMAGES statement
  • SYNC MEMORY statement
  • CRITICAL and END CRITICAL statements
  • LOCK and UNLOCK statements
  • ERROR STOP statement
  • ALLOCATE and DEALLOCATE may specify coarrays
  • Intrinsic procedures IMAGE_INDEX, LCOBOUND, NUM_IMAGES, 
    THIS_IMAGE, UCOBOUND
Note: ATOMIC_DEFINE and ATOMIC_REF are not supported in this version

- CONTIGUOUS attribute
- MOLD keyword in ALLOCATE
- DO CONCURRENT
- NEWUNIT keyword in OPEN
- G0 and G0.d format edit descriptor
- Unlimited format item repeat count specifier
- A CONTAINS section may be empty
- Intrinsic procedures BESSEL_J0, BESSEL_J1, BESSEL_JN, BESSEL_YN, BGE, BGT, BLE, BLT, DSHIFTL, DSHIFTR, ERF, ERFC, ERFC_SCALED, GAMMA, HYPOT, IALL, IANY, IPARITY, IS_CONTIGUOUS, LEADZ, LOG_GAMMA, MASKL, MASKR, MERGE_BITS, NORM2, PARITY, POPCNT, POPPAR, SHIFTA, SHIFTL, SHIFTR, STORAGE_SIZE, TRAILZ,
- Additions to intrinsic module ISO_FORTRAN_ENV: ATOMIC_INT_KIND, ATOMIC_LOGICAL_KIND, CHARACTER_KINDS, INTEGER_KINDS, INT8, INT16, INT32, INT64, LOCK_TYPE, LOGICAL_KINDS, REAL_KINDS, REAL32, REAL64, REAL128, STAT_LOCKED, STAT_LOCKED_OTHER_IMAGE, STAT_UNLOCKED

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

4.1 What’s New in Intel® MKL 10.3 Update 1
- PARDISO/DSS: Added true F90 overloaded API (see the Intel® MKL reference manual for more information)
- PARDISO: Improved the statistical reporting to be more reader friendly
- Sparse BLAS: Improved performance of ?BSRMM functions on the latest Intel® processors
- FFTs: Support for negative strides
- FFT examples: Added examples for split-complex FFTs in C and Fortran using both the DFTI and FFTW3 interfaces
- VML: Improved performance of real in-place Add/Sub/Mul/Sqr functions on systems supporting SSE2 and SSE3
- Poisson Library: Changed the default behavior of the Poisson library functions from sequential to threaded operation
- Bug fixes

4.2 What’s New in Intel® MKL 10.3

- BLAS
  - New functions for computing 2 matrix-vector products at once: [D/S]GEM2VU, [Z/C]GEM2VC
  - New functions for computing mixed precision general matrix-vector products: [DZ/SC]GEMV
  - New function for computing the sum of two scaled vectors: *AXPBY
Intel® AVX optimizations in key functions: SMP LINPACK, level 3 BLAS, DDOT, DAXPY

- LAPACK
  - New C interfaces for LAPACK supporting row-major ordering
  - Integrated Netlib LAPACK 3.2.2 including one new computational routine (*GEQRFP) and two new auxiliary routines (*GEQR2P and *LARFGP) and the earlier LAPACK 3.2.1 update
  - Intel® AVX optimizations in key functions: DGETRF, DPOTRF, DGEQRF

- PARDISO
  - Improved performance of factor and solve steps in multi-core environments
  - Introduced the ability to solve for sparse right-hand sides and perform partial solves—produces partial solution vector
  - Improved performance of the out-of-core (OOC) factorization step
  - Support for zero-based (C-style) array indexing
  - Zeros on the diagonal of the matrix are no longer required in sparse data structures for symmetric matrices
  - New ILP64 PARDISO interface allows the use of both LP64 and ILP64 versions when linked to the LP64 libraries
  - The memory required for storing files on the disk in OOC mode can now be estimated just after reordering

- Sparse BLAS
  - Format conversion functions now support all data types (single and double precision for real and complex data) and can return sorted or unsorted arrays

- FFTs
  - New MPI FFTW 3.3alpha1 wrappers cover new cluster functionality
  - Improved load-balancing of cluster FFTs provides improved performance
  - Intel AVX optimizations in all 1D/2D/3D FFTs
  - Improved performance of 2D and 3D mixed-radix FFTs for single and double precision data for all systems supporting the SSE4.2 instruction set
  - Support for split-complex data represented as two real arrays introduced for 2D/3D FFTs
  - Support for 1D complex-to-complex transforms of large prime lengths
  - Introduced Hybrid parallelism (MPI + OpenMP*) on cluster 1D complex transforms and increased performance on vector lengths which are a multiple of the number of MPI processes

- VML
  - A new function for computing \((ax+b)/(cy+d)\) where \(a, b, c,\) and \(d\) are scalars, and \(x\) and \(y\) are real vectors: \(v[s/d]LinearFrac()\)
  - Intel AVX optimizations for real functions
  - A new mode for setting denormals to zero, overflow support for complex vectors, and for every VML function a new function with an additional parameter for setting the accuracy mode

- VSL
  - A set of new Summary Statistics functions was added covering basic statistics, covariance and correlation, pooled, group, partial, and robust covariance/correlation, quantiles and streaming quantiles, outliers detection algorithm, and missing values support
    - Performance optimized algorithms: MI algorithm for support of missing values, TBS algorithm for computation of robust covariance, BACON algorithm for detection of outliers, ZW algorithm for
computation of quantiles (streaming data case), and 1PASS algorithm for computation of pooled covariance
  - Improved performance of SFMT19937 Basic Random Number Generator (BRNG)
  - Intel® AVX optimizations: MT19937 and MT2203 BRNGs
- Documentation: Product documentation is available in the Microsoft Help Viewer* 1.x format that integrates with Microsoft Visual Studio* 2010
- Added runtime dispatching dynamic libraries allowing link to a single interface library which loads dependent libraries dynamically at runtime depending on runtime CPU detection and/or library function calls
- A new directory structure has been established to simplify integration of Intel MKL with the Intel® Parallel Studio XE family of products and directories formerly designated as "em64t" are now designated by the "intel64" tag
- Intel® Itanium® architecture (IA-64) support is not included in this release. Intel® MKL 10.2 is the latest release for IA-64
- The sparse solver functionality has been fully integrated into the core Intel MKL libraries and the libraries with "solver" in the filename have been removed from the product

4.3 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/intel-mkl-kb/all

4.4 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.

4.5 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

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.

5 Intel® Parallel Debugger Extension
This section summarizes changes, new features and late-breaking news about this version of the Intel® Parallel Debugger Extension.

5.1 New Features
- Improved Data Sharing Detection
  - Support for OpenMP® 3.0
  - Support for Windows® OS synchronization functions
  - Improved data sharing detection analysis performance

5.2 Known Issues
- Coarray elements cannot be viewed.
- Fortran multi-dimensional arrays are not displayed correctly and are not accepted in filter expressions.
- Fortran complex types are not displayed correctly.
- Filters cannot be set on Fortran arrays with custom array bounds.
- If you are using Microsoft Visual Studio 2005, there are six Intel-specific exceptions that must be enabled manually. Select Debug > Exceptions, expand the Win32 Exceptions tree, and enable items:
  
  a1a01db0 Intel Parallel Debugger Extension Exception 0
  a1a01db1 Intel Parallel Debugger Extension Exception 1
  a1a01db2 Intel Parallel Debugger Extension Exception 2
  a1a01db3 Intel Parallel Debugger Extension Exception 3
  a1a01db4 Intel Parallel Debugger Extension Exception 4
  a1a01db5 Intel Parallel Debugger Extension Exception 5
This needs to be done once per project.

- Disabling the Intel Debugging exceptions during a debug session may cause Visual Studio (up to Visual Studio 2008, SP1) to hang.

- Use of the Intel Parallel Debugger Extension requires that the OpenMP library be linked dynamically, which is the default. If you wish to use the Parallel Debugger Extension, do not use `/Qopenmp-link:static` to specify static linking of the OpenMP Library.

- Be sure to enable the parallel debug instrumentation (switch `/debug:parallel`) before you start parallel debugging:
  
  Project > [project name] Properties > Configuration Properties >
  Fortran > Debugging > Enable Parallel Debug Checks: Yes
  (/debug:parallel). Otherwise, the debugger will not detect datasharing events nor break on re-entrant calls.

- If you are using Microsoft Visual Studio 2008 and debugging 64-bit applications, you must have Visual Studio 2008 Service Pack 1 installed.
  
  - You can debug 64-bit applications under Visual Studio 2005 and 2008 without Service Packs only if they are linked to the low memory area. If not linked to the low memory area, you will not see any events until the debuggee terminates. After termination, all events are displayed in the event window. In order to debug 64-bit applications properly, set the base address to 0x10000 in
    Project > Properties > Linker > Advanced.

- Function local or heap variables are displayed as “???” in the data sharing event window.

- The SSE Registers window does not work for 64-bit applications - the window shows “???”

- Filters on static local variables are not set correctly via context menu.

- Reentrant call detection stops in Disassembly view.

- The debugger extension windows remain empty when their placement is changed from "docked" to "floating". The workaround is to either keep them docked or to restart the debug session after the placement was changed.

- The debugger extension requires the application to be started from Visual Studio. It does not work when attaching to an existing process.

- Windows settings are restored to default (Hexadecimal) when the window is hidden or closed and reopened again.

5.3 Documentation
Intel Parallel Debugger Extension Documentation can be accessed via the Help menu of Microsoft Visual Studio or by pressing the function key F1 after activation of a Parallel Debugger.
Extension window. Help is also available by clicking the link “HTML version” inside debugger-documentation.htm.

6 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 go to:
http://www.intel.com/design/literature.htm

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. Go to:

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.