Intel® Parallel Studio XE 2015 Composer Edition for Fortran Windows* Installation Guide and Release Notes

6 August 2015

Table of Contents

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. For the most current update to these release notes, see the release notes posted at the Intel® Software Development Products Registration Center where you downloaded this product.

Due to the nature of this comprehensive integrated software development tools solution, different Intel® Parallel Studio XE Composer Edition components may be covered by different licenses. Please see the licenses included in the distribution as well as the Disclaimer and Legal Information section of these release notes for details.

1.1 Changes in Update 5

- Intel® Visual Fortran Compiler updated to version 15.0.4
- Intel® Math Kernel Library 11.2 Update 4
- Support for Microsoft Windows 10*
- Support for Microsoft Visual Studio 2015*
- Corrections to reported problems

1.2 Changes in Update 4

- Corrections to reported problems in Installation and IDE integration

1.3 Changes in Update 3

- Intel® Visual Fortran Compiler updated to version 15.0.3
- Intel® Math Kernel Library 11.2 Update 3
- Corrections to reported problems

1.4 Changes in Update 2

- New Library Support for Assignment with Allocatable Polymorphic Components
- Microsoft Visual Studio Community 2013 supported

- Intel® Visual Fortran Compiler updated to version 15.0.2
- Intel® Math Kernel Library 11.2 Update 2
- Corrections to reported problems

1.5 Changes in Update 1

- Support for Intel® Advanced Vector Extensions 512 instructions for IA-32 and Intel® 64 architectures in 15.0.1
- Tools->Options and Project Menu Labels Changed in 2015 Update 1
- First update with Japanese Localization
- New Fortran Project from Existing Code
- MIN/MAX Reductions supported in SIMD Loop Directive
- Intel® Visual Fortran Compiler updated to version 15.0.1
- Intel® Math Kernel Library 11.2 Update 1
- Corrections to reported problems

1.6 Changes since Intel® Visual Fortran Composer XE 2013 SP1 (New in Intel® Parallel Studio XE 2015 Composer Edition)

- Intel® Visual Fortran Compiler updated to version 15.0
    - New Optimization Report interface, structure, and options (users of existing options /Qopt-report, /Qvec-report, /Qopenmp-report, and /Qpar-report are strongly encouraged to consult the Intel Compiler User's Guide for additional details)

- New IDE integration for optimization reports showing report information integrated with source with hyperlinking to relevant areas. See the User's Guide for details.
- Additional OpenMP* 4.0 support
- Select custom installation configurations with the online installer
- Enable threadsafe profile generation with PGO
- New INTEL_PROF_DYN_PREFIX environment variable to add custom prefix to PGO .dyn filenames
- Static Analysis is deprecated
- Windows XP* not supported
- Microsoft Visual Studio 2008* not supported
- Intel® Debugger Extension for Intel® MIC Architecture updated to version 7.7-8.0
- Intel® Math Kernel Library updated to version 11.2
- Corrections to reported problems

1.7 Product Contents

*Intel® Parallel Studio XE 2015 Update 5 Composer Edition for Fortran Windows** includes the following components:

- Intel® Visual Fortran Compiler XE 15.0.4 for building applications that run on IA-32 and Intel® 64 architecture systems
- Intel® Math Kernel Library 11.2 Update 4
- Intel® Debugger Extension for Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
- Fortran Expression Evaluator (FEE) for debugging Fortran applications with Microsoft Visual Studio*
- Integration into Microsoft* development environments
- Microsoft Visual Studio 2010* Shell and Libraries (not included with Evaluation licenses)
- Sample programs
- On-disk documentation

1.8 System Requirements

For an explanation of architecture names, see 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
- 2GB RAM (4GB recommended)
- 4GB free disk space required for all product features and all architectures
- For Intel® Many Integrated Core Architecture (Intel® MIC Architecture) development/testing:
    - Intel® Xeon Phi™ processor
    - Intel® Manycore Platform Software Stack (Intel® MPSS)
    - Debugging of offload code requires Microsoft Visual Studio* 2012 or 2013
- Microsoft Windows 7 Professional*, Microsoft Windows 8*, Microsoft Windows 8.1*, Microsoft Windows 10*, Microsoft Windows Server 2012* (R2), Microsoft Windows Server 2008 SP2* (IA-32 architecture only), Microsoft Windows Server 2008 R2 SP1 (Intel® 64 architecture only) or Microsoft Windows HPC Server 2008 (R2)* (embedded editions not supported)
    - Microsoft Windows Server 2008 or Windows HPC Server 2008 requires Microsoft Visual Studio 2013* or Visual Studio 2012* or Visual Studio 2010* or Visual Studio 2010* Shell.
    - On Microsoft Windows 8, Microsoft Windows 8.1, and Microsoft Windows Server 2012, the product installs into the “Desktop” environment. Development of “Windows 8* UI” applications is not supported.
- To use the Microsoft Visual Studio development environment or command-line tools to build IA-32 or Intel® 64 architecture applications, one of:
    - Microsoft Visual Studio 2015* Professional Edition (or higher edition) with 'Common Tools for Visual C++ 2015' component installed
    - Microsoft Visual Studio Community 2015* with 'Common Tools for Visual C++ 2015' component installed
    - Microsoft Visual Studio 2013* Professional Edition or higher

    - Microsoft Visual Studio Community 2013
    - Microsoft Visual Studio 2012* Professional Edition or higher
    - Microsoft Visual Studio 2010* Professional Edition or higher, with C++ component installed
    - Intel® Visual Fortran development environment based on Microsoft Visual Studio 2010 Shell (included with some license types of Intel® Fortran Compiler) [1]
- To use command-line tools only to build IA-32 architecture applications, one of:
    - Microsoft Visual Studio Express 2013 for Windows Desktop*
    - Microsoft Visual Studio Express 2012 for Windows Desktop*
    - Microsoft Visual C++ 2010* Express Edition [2]
- To use command-line tools only to build Intel® 64 architecture applications, one of:
    - Microsoft Visual Studio Express 2013 for Windows Desktop*
    - Microsoft Visual Studio Express 2012 for Windows Desktop*
    - Microsoft Windows Software Development Kit for Windows 8.1*
    - Microsoft Windows Software Development Kit for Windows 8*
- To read the on-disk documentation, Adobe Reader* 7.0 or later

Notes:

1. Intel® Visual Fortran development environment based on Microsoft Visual Studio 2010* Shell is included with Academic and Commercial licenses for Intel® Parallel Studio XE 2015 Composer Edition for Fortran Windows*. It is not included with Evaluation 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*
2. Microsoft Visual C++ 2010 Express Edition will coexist with the Intel® Parallel Studio XE 2015 Composer Edition for Fortran Windows installation of Microsoft Visual Studio 2010 Shell. Note that the C++ and Fortran development environments will be separate.
3. 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. Note, however, that applications calling Intel® MKL require a processor supporting the Intel® SSE2 instructions.
4. 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 7, though Intel does not test these for compatibility. Your application may depend on a Windows 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.

1.8.1 Visual Studio 2008* is Not Supported

Support has been removed for installation and use with Visual Studio 2008. Intel recommends migrating to a newer version of Visual Studio*.

1.8.2 Windows XP* is Not Supported

Support has been removed for installation and use on Windows XP. Intel recommends migrating to a newer version of these operating systems.

1.9 Documentation

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

1.9.1 Documentation on Creating Windows-based Applications on the Web

Documentation on using QuickWin, dialogs and the Windows API is available from the Intel Software Documentation Library: See Using Intel® Visual Fortran to Create and Build Windows*-based Applications (PDF)

1.9.2 Documentation Viewing Issue with Microsoft Internet Explorer* 10 and Windows Server* 2012

If on Windows Server 2012 you find that you cannot display help or documentation from within Internet Explorer 10, modifying a security setting for Microsoft Internet Explorer usually corrects the problem. From Tools > Internet Options > Security, add “about:internet” to the list of trusted sites. Optionally, you can remove “about:internet” from the list of trusted sites after you finish viewing the documentation.

1.10 Optimization Notice

Optimization Notice

Intel’s compilers 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 SSE2, SSE3, and 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. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

1.11 Samples

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

1.12 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.

Japanese language support will be available in an update on or after the release of Intel® Parallel Studio XE 2015 Composer Edition for Fortran Windows.

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 Changing Language Setting to see English on a Japanese OS Environment or Vice Versa on Windows.

1.13 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 Install Prerequisite Software

If you will be installing the included Microsoft Visual Studio 2010 Shell, additional Microsoft software may be required to be installed before beginning the installation of Intel® Parallel Studio XE 2015 Composer Edition for Fortran Windows*.

Microsoft .NET 4.0 Framework* is required. If you do not already have this installed, you can download the installer:

- .NET 4.0 Framework 32-bit and 64-bit

If you are installing on Windows 8.1*, Windows 8*, Windows 7* or Windows Server 2008, the installation of the Shell will attempt to download and install .NET Framework 4.0 automatically if it is not already present. If this fails, the Intel® Parallel Studio XE 2015 Composer Edition for Fortran Windows* install will fail with a message that may not indicate the exact problem. If you

find that the installation of the Shell fails, please download .NET 4.0 Framework from the above link and try again.

If you are installing Intel® Parallel Studio XE 2015 Composer Edition for Fortran Windows* from the full product downloadable that includes Visual Studio 2010 Shell, it will try to install Visual Studio 2010 Shell if you do not already have Visual Studio 2010 installed. If you do not want Visual Studio 2010 Shell to install, you can choose a Custom install and deselect it.

2.2 Installation of Intel® Manycore Platform Software Stack (Intel® MPSS)

The Intel® Manycore Platform System Software (Intel® MPSS) should be installed only if you plan to build applications that target an Intel® Xeon Phi™ coprocessor. Intel® MPSS may be installed before or after installing the Intel® Parallel Studio XE 2015 Composer Edition for Fortran Windows* product.

Using the latest version of Intel® MPSS available is recommended. It is available from the Intel® Software Development Products Registration Center at http://registrationcenter.intel.com as part of your Intel® Parallel Studio XE for Windows* registration. Refer to the Intel® MPSS documentation for the necessary steps to install the user space and kernel drivers.

2.3 Online Installer

The default electronic installation package now consists of a smaller installation package that dynamically downloads and then installs packages selected to be installed. This requires a working internet connection and potentially a proxy setting if you are behind an internet proxy. Full packages are provided alongside where you download this online install package if a working internet connection is not available. The online installer may be downloaded and saved as an executable file which can then be launched from the command line.

2.3.1 Storing Online Installer Download Content

The online installer stores the downloaded content in the form-factor of the standard install package which can then be copied and reused offline on other systems. The default download location is <ProgramFiles>\intel\downloads. This location may be changed with the online installer command line option "--download-dir [FOLDER]". The online installer also supports a download only mode which allows the user to create a package without installation. This mode is enabled with the "--download-only" command line option.

2.4 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.

After downloading the product, 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.

Register your serial number at the Intel® Software Development Products Registration Center for access to product updates and previous versions.

2.4.1 Reboot After Install Recommended

Installation of Intel® Parallel Studio XE 2015 Composer Edition for Fortran Windows* adds to the system PATH environment variable the folders containing the compiler run-time DLLs (but not those of Intel® Math Kernel Library). On some systems, if the length of the PATH value is between 2048 and 4096 characters, command line operations may fail until the system is rebooted. Intel recommends a reboot after the first install of Intel® Parallel Studio XE 2015 Composer Edition for Fortran Windows*.

2.4.2 Cluster Installation

If Microsoft Compute Cluster Pack* is present, and the installation detects that the installing system is a member of a cluster, the product will be installed on all visible nodes of the cluster when a “Full” installation is requested. If a “Custom” installation is requested, you will be given the option to install on the current node only.

2.4.3 Using a License Server

If you have purchased a “floating” license, see Licensing: Setting Up the Client for a Floating License. This article also provides a source for the Intel® License Manager for FLEXlm* product that can be installed on any of a wide variety of systems.

2.4.4 Additional Steps to Install Documentation for Microsoft Visual Studio 2010*

When installing Intel® Parallel Studio XE 2015 Composer Edition for Fortran Windows* 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® Parallel Studio XE 2015 Composer Edition for Fortran Windows* help documentation within Visual Studio 2010. Please follow the instructions of the "Help Library Manager" installation wizard to install the Intel® Parallel Studio XE 2015 Composer Edition for Fortran Windows* help documentation for Visual Studio 2010.

This step is only needed once. When you install Intel® Parallel Studio XE 2015 Composer Edition for Fortran Windows* updates in the future, you will not be required to re-register the documentation through the “Help Library Manager”.

For more information, see http://msdn.microsoft.com/en-us/subscriptions/dd264831(v=vs.100).aspx or search on microsoft.com for “Help Library Manager”.

2.4.5 Using the Intel® Visual Fortran Compiler with Microsoft Visual Studio 2015*

To use the Intel® Visual Fortran Compiler with Microsoft Visual Studio 2015*, it is necessary to install the 'Common Tools for Visual C++ 2015' component from Visual Studio. This article explains how.

2.5 Intel® Software Manager

The installation provides the Intel® Software Manager to provide a simplified delivery mechanism for product updates and provide current license status and news on all installed Intel® software products.

You can also volunteer to provide Intel anonymous usage information about these products to help guide future product design. This option, the Intel® Software Improvement Program, is not enabled by default – you can opt-in during installation or at a later time, and may opt-out at any time. For more information please see Intel® Software Improvement Program.

2.6 Changing, Updating and Removing the Product

Use the Windows Control Panel "Add or Remove Products" or "Programs and Features" 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® Parallel Studio XE 2015 Composer Edition for Fortran Windows*
- Intel® Parallel Studio XE 2015 Composer Edition for Windows*
- Intel® Parallel Studio XE 2015 Professional Edition for Windows*
- Intel® Parallel Studio XE 2015 Professional Edition for Fortran Windows*

If you also installed Microsoft Visual Studio 2010 Shell as part of the compiler install, the following additional entries may be present:

- Microsoft Visual Studio 2010 Shell (Integrated) - ENU
- Microsoft Visual Studio 2010 Files for Intel Visual Fortran
- Microsoft Visual Studio 2010 Remote Debugger – ENU

The entries for Visual Studio Shell, Files and Remote Debugger 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 Composer XE > Visual Fortran Compiler dialog. Compiler versions older than 13.0 (Intel® Visual Fortran Composer XE 2011) 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.

2.7 Silent Install and Uninstall

For information on how to install and uninstall the compiler in an automated fashion, please see Intel® Compilers for Windows* Silent Installation Guide.

2.7.1 Support of Non-Interactive Custom Installation

Intel® Parallel Studio XE 2015 supports the saving of user install choices during an 'interactive' install in a configuration file that can then be used for silent installs. This configuration file is created when the following option is used from the command line install:

- `--duplicate=config_file_name`: it specifies the configuration file name. If the full path name is specified, "`--download-dir`" is ignored and the installable package will be created under the same directory as the configuration file.
- `--download-dir=dir_name`: optional, it specifies where the configuration file will be created. If this option is omitted, the installation package and the configuration file will be created under the default download directory:

```
%Program Files%\Intel\Download\<package_id>
```

For example:

```
w_fcompxe_online_2015.0.0XX.exe --duplicate=ic15_install_config.ini
  --download-dir "C:\temp\custom_pkg_ic15"
```

The configuration file and installable package will be created under "`C:\temp\custom_pkg_ic15`".

2.8 Installation Folders

The installation folder arrangement is shown in the diagram below. Not all folders will be present in a given installation. The system environment variable `IFORT_COMPILER15` can be used to locate the most recently installed Intel® Parallel Studio XE 2015 Composer Edition for Fortran Windows*.

- `C:\Program Files\Intel\Composer XE 2015`
  - `bin`
    - `ia32`
    - `ia32_intel64`
    - `intel64`
    - `intel64_mic`
    - `sourcechecker`
  - `compiler`
    - `include`
      - `ia32`
      - `intel64`
      - `mic`
    - `lib`
      - `ia32`

            - intel64
            - mic
    - debugger
    - Documentation
    - mkl
        - benchmarks
        - bin
        - examples
        - include
        - interfaces
        - lib
        - tests
        - tools
    - redist
    - Samples

Where the folders under `bin`, `include` and `lib` are used as follows:

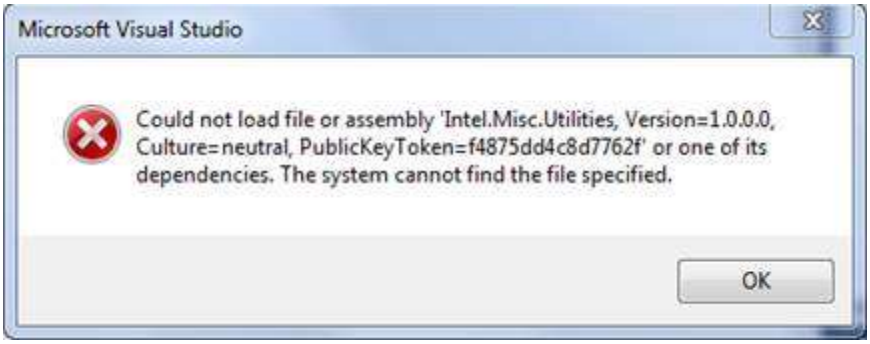
- `ia32`: Files used to build applications that run on IA-32 architecture
- `intel64`: Files used to build applications that run on Intel® 64
- `ia32_intel64`: Compilers that run on IA-32 architecture to build applications that run on Intel®64

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 `Composer XE 2015.`*`nnn`* where *`nnn`* is the update number.

2.9 Known Installation Issues and changes

- Remote offline activation via a code has been removed. Use of a license file or a license manager remain as options for alternative activation.
- When installing “Intel® Parallel Studio XE 2015 Composer Edition for Fortran Windows*” on a system which has “Intel(R) C++ Composer XE 2013 SP1 Update 2” installed you may get the following error in Visual Studio: “Could not load file or assembly ‘Intel.Misc.Utilities’ or one of its dependencies. The system cannot find the file specified.”.



This is due to an issue in “Intel(R) C++ Composer XE 2013 SP1 Update 2”, which was fixed in subsequent releases. This can be resolved by installing “Intel(R) C++ Composer XE 2013 SP1 Update 3” or “Intel® Parallel Studio XE 2015 Composer Edition for C++ Windows*”.

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 15.0. Exceptions include:

- Sources that use the CLASS keyword to declare polymorphic variables and which were built with a compiler version earlier than 12.0 must be recompiled.
- Objects built with the multi-file interprocedural optimization (`/Qipo`) option must be recompiled.
- Objects that use the REAL(16) , REAL*16, COMPLEX(16) or COMPLEX*32 datatypes and which were compiled with versions earlier than 12.0 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 outside of a derived type declaration and were compiled with versions earlier than 11.0 must be recompiled. The compiler will notify you if this issue is encountered.
- Modules that specified an ATTRIBUTES ALIGN directive inside a derived type declaration cannot be used by compilers older than 13.0.1.

3.1.1 Stack Alignment Change for REAL(16) and COMPLEX(16) Datatypes (12.0)

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,

compiler versions 12.0 and later align such items at 16 bytes and expect 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 13 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.

3.1.2 Static Form of the Intel® OpenMP* Library is No Longer Provided

The static form of the Intel® OpenMP* library, libiomp5mt.lib, is no longer provided, and the `/Qopenmp-link:static` command line option is no longer supported. Please replace all references to libiomp5mt.lib with libiomp5md.lib, the DLL import library. This change also implies that applications using OpenMP will need to have the Intel® compiler redistributables installed if deployed on a system where an Intel® compiler is not also present. See Redistributable Libraries for Intel® Parallel Studio XE for more information.

3.1.3 Fortran Expression Evaluator

Fortran Expression Evaluator (FEE) is a plug-in for Microsoft Visual Studio* that is installed with Intel® Visual Fortran Compiler. It extends the standard debugger in Microsoft Visual Studio* IDE by handling Fortran expressions. There is no other change in usability.

3.1.4 New Library Support for Assignment with Allocatable Polymorphic Components (15.0.2)

In order to resolve incorrect behavior in some circumstances when an assignment is done, either explicitly or implicitly, of a derived type containing components that are allocatable and polymorphic, a new compiler support library routine was created for version 15.0.2 and is called by the generated code. This means that if a Fortran source that has such an assignment is compiled with the 15.0.2 (or newer) compiler, it must be linked with the Fortran libraries from 15.0.2 or later. Failure to do this may lead to link errors regarding a missing routine `for_alloc_assign_v2`. In general you should always link with a library version no older than the compiler.

3.2 New and Changed Compiler Features

Some language features may not yet be described in the compiler documentation. Please refer to the Fortran 2003 Standard (PDF)) and Fortran 2008 Standard (PDF) if necessary.

3.2.1 Features from Fortran 2003

- Parameterized Derived Types

3.2.2 Features from Fortran 2008

- BLOCK construct
- intrinsic subroutine EXECUTE_COMMAND_LINE

3.2.3 Features from OpenMP*

The following directives, clauses and procedures, from OpenMP 4.0, are supported by the compiler. Some of these features were supported in Intel® Visual Fortran Composer XE 2013 Update 3 based on a preliminary specification, some keywords supported earlier (DECLARE TARGET MIRROR, DECLARE TARGET LINKABLE, MAPTO, MAPFROM, SCRATCH) are no longer supported, and some syntax has changed its meaning since the earlier specification.

For more information, see the compiler documentation or the link to the OpenMP Specification above.

SIMD Directives:

- OMP SIMD
- OMP DECLARE SIMD
- OMP DO SIMD
- OMP PARALLEL DO SIMD

Coprocessor Directives:

- OMP TARGET DATA
- OMP TARGET
- OMP TARGET UPDATE
- OMP DECLARE TARGET

Other Directives:

- OMP PARALLEL PROC_BIND
- OMP TASKGROUP
- OMP CANCEL
- OMP CANCELLATION POINT

Clauses:

- MAP
- DEPEND

Procedures:

- OMP_GET_DEVICE_NUM
- OMP_GET_PROC_BIND
- OMP_SET_DEVICE_NUM

3.2.3.1 KMP_PLACE_THREADS Environment Variable (13.1.0)

This environment variable allows the user to simplify the specification of the number of cores and threads per core used by an OpenMP application, as an alternative to writing explicit affinity settings or a process affinity mask.

3.2.3.2 KMP_DYNAMIC_MODE Environment Variable Support for "asat" Deprecated

Support for "asat" (automatic self-allocating threads) by the environment variable KMP_DYNAMIC_MODE is now deprecated, and will be removed in a future release.

3.2.4 New and Changed Directives

The following compiler directives are new or changed in Intel® Parallel Studio XE 2015 Composer Edition– please see the documentation for details:

- ATTRIBUTES OPTIMIZATION_PARAMETER INLINE-MAX-TOTAL-SIZE=N
- ATTRIBUTES OPTIMIZATION_PARAMETER INLINE-MAX-PER-ROUTINE=N

3.2.4.1 ATTRIBUTES STDCALL now allowed with BIND(C)

As of compiler version 15.0, the ATTRIBUTES STDCALL directive may be specified for an interoperable procedure (a procedure whose declaration includes the BIND(C) language binding attribute.) This combination has the following effects for Windows* applications targeting IA-32 architecture:

- The calling mechanism is changed to STDCALL, which affects how the stack is cleaned up on procedure exit
- The external name from the BIND attribute is suffixed with "@n", where n is the number of bytes to be removed from the stack on return.

No other effects from STDCALL, such as pass-by-value, are provided. The Fortran standard VALUE attribute (not ATTRIBUTES VALUE) may be used if desired. For all other platforms, specifying STDCALL with BIND(C) has no effect.

3.2.5 Other Features

For information on these features, please see the compiler documentation.

- New environment variable INTEL_PROF_DYN_PREFIX. Allows the user to have some control over the naming PGO generated ".dyn" files to make it easy to distinguish files from different runs. By setting this environment variable to the desired character string prior to starting the instrumented application, the string will be included as prefix to the .dyn file names.
- New intrinsic __intel_simd_lane() that represents the 'lane id' within a SIMD vector. This feature supports writing short-vector hyperobject reducer implementation. It also enables the performing of reduction operations inside SIMD-enabled functions.

3.2.6 Coarrays (13.0)

No special procedure is necessary to run a program that uses coarrays in a shared-memory environment; 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.

A license for Intel® Parallel Studio XE Cluster Edition must be present in order to use the `/coarray:distributed` option. For details on how to run a distributed coarray application on Windows, please see Building and Running a Distributed Coarray Application on Windows.

Use of coarray applications with any MPI implementation other than Intel® MPI, or with OpenMP*, is not supported at this time.

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`.

3.2.6.1 Coarrays and Intel® MPI Library compatibility

Coarrays with Intel® Fortran Compiler 14 is incompatible with Intel® MPI Library 5.0. If using coarrays, ensure that either Intel® Fortran Compiler 15 or higher is used, or use a 4.x version of Intel® MPI Library.

3.2.7 ATTRIBUTES ALIGN for component of derived type (13.0.1)

As of compiler version 13.0.1, the ATTRIBUTES ALIGN directive may be specified for an ALLOCATABLE or POINTER component of a derived type. The directive must be placed within the derived type declaration, and if it is an extended type, the directive must not name a component in a parent type.

If this is specified, the compiler will apply the indicated alignment when the component is allocated, either through an explicit ALLOCATE or, for ALLOCATABLE components, through implicit allocation according to Fortran language rules.

A module containing an ATTRIBUTES ALIGN directive for a derived type component cannot be used with a compiler earlier than version 13.0.1.

3.2.8 Change in File Buffering Behavior (13.1)

In product versions prior to Intel® Visual Fortran Composer XE 2013 (compiler version 13.0), the Fortran Runtime Library buffered all input when reading variable length, unformatted sequential file records. This default buffering was accomplished by the Fortran Runtime Library allocating an internal buffer large enough to hold any sized, variable length record in memory. For extremely large records this could result in an excessive use of memory, and in the worst cases could result in available memory being exhausted. The user had no ability to change this default buffering behavior on such READs. There was always the ability to request or deny buffering of these records when writing them, but not when reading them.

This default buffering behavior was changed with the release of Intel® Visual Fortran Composer XE 2013. Beginning with this version, all such records are **not** buffered by default, but rather read directly from disk to the user program's variables. This change helped programs that needed to conserve memory, but could in fact result in a performance degradation when reading records that are made of many small components. Some users have reported this performance degradation.

The Intel® Visual Fortran Composer XE 2013 Update 2 (compiler version 13.1) release of the Fortran Runtime Library now provides a method for a user to choose whether or not to buffer these variable length, unformatted records. The default behavior remains as it was in 13.0; these records are not buffered by default. If you experience performance degradation when

using 13.1 with this type of I/O, you can enable buffering of the input the same way that you choose whether to enable buffering of the output of these records – one of the following:

- specifying BUFFERED=”YES” on the file's OPEN statement
- specifying the environment variable FORT_BUFFERED to be YES, TRUE or an integer value greater than 0
- specifying `-assume buffered_io` on the compiler command line

In the past, these mechanisms applied only when issuing a WRITE of variable length, unformatted, sequential files. They can now be used to request that the Fortran Runtime Library buffer all input records from such files, regardless of the size of the records in the file.

Using these mechanisms returns the READing of such records to the pre-13.0 behavior.

3.2.9 Static Analysis has been deprecated

Static analysis is deprecated. It may be removed in a future major release. If you have concerns or feedback, please comment.

3.2.10 New run-time routines to get Fortran library version numbers

- FOR_IFCORE_VERSION returns the version of the Fortran run-time library (ifcore).
- FOR_IFPORT_VERSION returns the version of the Fortran portability library (ifport).

3.2.11 Support for Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instructions for IA-32 and Intel® 64 architectures in 15.0.1

The Intel® Compiler 15.0.1 now supports Intel® AVX-512 instructions for processors based on IA-32 and Intel® 64 architectures that support that instruction set. The instructions are supported via inline assembly, and/or the /Q[a]xCORE-AVX512 compiler options. This is in addition to the current support for Intel® AVX-512 instructions for Intel® Many Integrated Core Architecture.

3.2.12 MIN/MAX Reductions supported in SIMD Loop Directive

Starting with the Intel® Compilers version SIMD Loop Directive now supports MIN/MAX reductions, like so:

```
!DIR$ SIMD REDUCTION(MAX:SIMDMAX)
      DO I = 1, SIZE
         IF (X(I) > SIMDMAX) SIMDMAX = X(I)
      END DO

!DIR$ SIMD REDUCTION(MIN:SIMDMIN)
      DO I = 1, SIZE
         IF (X(I) < SIMDMIN) SIMDMIN = X(I)
      END DO

!DIR$ SIMD REDUCTION(MAX:XMAX)
      DO I = 1, SIZE
         XMAX = MAX (XMAX, X(I))
      END DO
```

```
!DIR$ SIMD REDUCTION(MIN:XMIN)
      DO I = 1, SIZE
         XMIN = MIN (XMIN, X(I))
      END DO
```

3.3 New and Changed Compiler Options

Please refer to the compiler documentation for details.

3.3.1 New and Changed in Intel® Parallel Studio XE 2015 Composer Edition for Fortran Windows*

- /assume:[no]std_value
- /assume:ieee_fpe_flags
- /fast
- /Qeliminate-unused-debug-types[-]
- /Qinit:snan
- /Qopt-dynamic-align[-]
- /Qopt-report
- /Qprof-gen:[no]threadsafe

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

3.3.1.1 /assume:std_value is now the default

As of compiler version 15.0, the Fortran standard VALUE attribute, (not ATTRIBUTES VALUE), when specified for a dummy argument of a non-interoperable procedure (a procedure whose declaration does not include the BIND(C) language binding attribute), applies Fortran standard semantics by default. The standard specifies that for a non-interoperable procedure, VALUE causes a temporary, redefinable copy of the actual argument to be passed using the default passing mechanism. In earlier compiler versions, VALUE always caused the actual argument to be passed by value. Compiler version 14.0 introduced /assume:std_value to specify the standard-conforming semantics and this was enabled if /standard-semantics was specified.

3.3.1.2 /assume:ieee_fpe_flags enabled with /standard-semantics and /fp:strict or /fp:except

As of compiler version 15.0, if /standard-semantics and one of /fp:strict or /fp:except is specified, /assume:ieee_fpe_flags is also enabled. This option causes the state of floating point exceptions to be saved on entry to a procedure and restored on exit. The save and restore operation has a significant performance penalty so it should be used only by applications that manipulate or query the floating point exception environment. Note that Intel Fortran requires that you specify /fp:strict if you are using the Fortran standard intrinsic modules IEEE_ARITHMETIC, IEEE_EXCEPTIONS and/or IEEE_FEATURES.

3.3.1.3 Change to /fast option

/fp:fast=2 has been added to the /fast option. This option makes it easier to tune for performance.

3.3.1.4 New /Qinit:snan Compiler Option

A new command line option to help find a class of uninitialized variables at run-time by initializing floating-point variables to signaling NaNs which can then be trapped if their values are fetched before being set.

3.3.1.5 New /Qopt-dynamic-align[-] Compiler Option

When this option is set the compiler implements conditional optimizations based on dynamic alignment of the input data for maximum performance of vectorized code especially for long trip count loops. This, however, may result in different bitwise results for aligned and unaligned data with the same values. When unset the compiler will not perform these optimizations providing bitwise reproducibility.

3.3.1.6 New Optimization Report interface, report structure, and options in Intel® Parallel Studio XE 2015 Composer Edition

The four kinds of optimization reports (/Qopt-report, /Qvec-report, /Qopenmp-report, and /Qpar-report) have been consolidated under one /Qopt-report interface in this version of Intel® Parallel Studio XE 2015 Composer Edition. This consolidated optimization report has been rewritten to improve the presentation, content, and precision of the information provided so that users better understand what optimizations were performed by the compiler and how they may be tuned to yield the best performance.

The output of this report no longer defaults to stderr due to issues with parallel builds. Instead, by default an output file (extension .optrpt) containing the report for each corresponding source file is created in the target directory of the compilation process (i.e. the same directory where object files would be generated). /Qopt-report-file (for example: /Qopt-report-file:stderr) can be used to change this behavior.

The /Qvec-report, /Qopenmp-report, and /Qpar-report options have been deprecated, but they remain and map to corresponding values of the /Qopt-report option. However, the report information and formatting, and the default to reporting to a file, will follow the new opt-report model.

It is strongly recommended that you read the documentation for full details. See the Intel Compiler User's Guide under `Compiler Reference->Compiler Option Categories and Descriptions->Optimization Report Options`.

3.3.1.7 New mode for PGO instrumentation /Qprof-gen:[no]threadsafe

This change adds a mode to the PGO instrumentation that allows for the collection of PGO data on applications that use a high level of parallelism, such as from OpenMP 3.1. This functionality improves PGO usage for the IA-32 and Intel® 64 architectures, as well as enables the support of PGO with the native Intel® MIC Architecture programming model.

3.4 Visual Studio Integration Changes

3.4.1 DLL Libraries Default in New Projects (14.0)

New Fortran projects, created after Intel® Parallel Studio XE 2015 Composer Edition for Fortran Windows*has been installed, have the project properties set so that the DLL form of the run-time libraries is used. This is consistent with Microsoft Visual C++, but is a change from previous versions of Intel® Visual Fortran. If you wish to use the static libraries, you can change the project property Fortran > Libraries > Use Runtime Library. Note that the OpenMP* library, libiomp5md.dll, is provided in DLL form only and will be used no matter which setting you select, should your application use OpenMP.

3.4.2 Parallel Build Option (13.1)

An enhancement to the Visual Studio build environment has been added which allows for parallel builds of sources without unresolved dependencies on multicore or multiprocessor systems. This can reduce the total time needed to build larger projects.

To enable this, open the project property page Fortran > General and set the property “Multi-processor compilation” to Yes.

3.4.3 Improved source code navigation in Microsoft Visual Studio IDE

The Visual Studio IDE now provides a “tree-view” for easy module/procedure navigation (similar to the Solution explorer view). For more information, see the compiler documentation.

3.4.4 Changes in Optimization Report Options support in Microsoft Visual Studio IDE

Optimization Diagnostic Level, Optimization Diagnostic Phase, Vectorizer Diagnostic Level, OpenMP Diagnostic Level, Auto-Parallelizer diagnostic Level project properties’ values (Configuration Properties->Fortran->Diagnostics) were updated in Intel® Parallel Studio XE 2015 Composer Edition for Fortran Windows*. If you are using these properties, you may need to update their values, using project property pages dialog in Visual Studio. If you change your settings to use older compiler, you may need to update these properties’ values again.

3.4.5 Changes in Online Help format in Microsoft Visual Studio*

The online help format is now browser-based. When you view Intel documentation from the Microsoft Visual Studio Help menu, or when you view context-sensitive help using F1 or a help button in a dialog box or other GUI element, your default browser shows the corresponding help topic. You may encounter some minor functionality issues depending on your default browser. Known issues include:

- When Set Help Preference is set to Launch in Browser and you hit F1 in Tools>Options>F# Tools or Tools>Options>Intellitrace**,** the browser appears twice.
- Chrome*: When arriving at a topic from Search or Index, the Table of Contents (TOC) does not sync, nor does the Sync TOC link work.
- Firefox*: The TOC loses context easily. Search is case sensitive

- Safari*: Response on Windows is slow.

3.4.6 Tools->Options and Project Menu Labels Changed in 2015 Update 1

Beginning in Intel® Parallel Studio XE 2015 update 1, some of the labels used to identify the Intel® Compiler have changed. Specifically:

- Under the `Tools->Options` menu, the label “Intel Parallel Studio XE” on the left is now called “Intel Compilers and Tools”. The settings available to be set there have not changed (for example, include directories, Code Coverage settings, or Performance Libraries settings).
- Under the `Project` menu, or when you right-click on a project, the menu entry “Intel Compiler XE 15.0” is now called “Intel Compiler”.

3.4.7 New Fortran Project from Existing Code

In Visual Studio you can now select File > New > Fortran Project From Existing Code. This will create a new Fortran project with sources added from a folder you select. The project wizard will allow you to customize the project type and platform.

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.

3.5.2 Debugging might fail when only Microsoft Visual Studio 2010/2013/2015 is installed

On Microsoft Windows* systems with only Microsoft Visual Studio 2010/2013/2015* installed debugging of Fortran applications might fail. Some symptoms might be failing watches (expression evaluations) or conditional breakpoints.

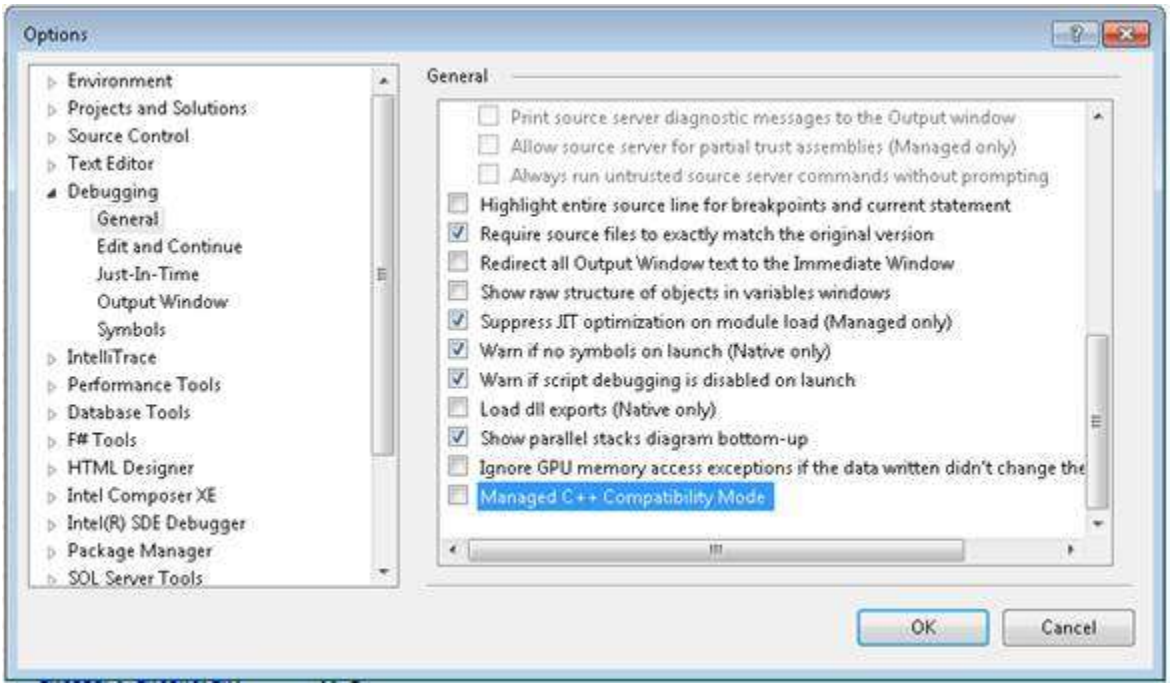
Intel® Parallel Studio XE 2015 Composer Edition for Fortran Windows* provides a debugger extension called Fortran Expression Evaluator (FEE) to enable debugging of Fortran applications. For some FEE functionality the Microsoft Visual Studio 2012* libraries are required. One solution is to install Microsoft Visual Studio 2012* in addition to Microsoft Visual Studio 2010/2013/2015*. An alternative is to install the "Visual C++ Redistributable for Visual Studio 2012 Update 4" found here.

If you installed Intel® Parallel Studio XE 2015 (with Intel® Composer XE 2015 Update 4 or later) on a system without any Microsoft Visual Studio* version available, a Microsoft Visual Studio 2010* Shell (incl. libraries) will be installed. It might be that FEE does not work in that environment. Please install the redistributable package mentioned above in addition to enable FEE. A future update will solve this problem for the installation of the shell.

3.5.3 Debugging mixed language programs with Fortran does not work

To enable debugging Fortran code called from a .NET managed code application in Visual Studio 2012 or later, unset the following configuration:

Menu Tools ->Options, under section Debugging->General, clear the Managed C++ Compatibility Mode check box



For any managed code application, one must also check the project property Debug > Enable unmanaged code debugging.

3.5.4 Update for Windows 7 causes LNK1123 error when building Visual Studio 2010 projects

A Microsoft Windows Update for .NET Framework 4.5.1, or installing Visual Studio 2012, installs a new version of a DLL used in the linking process, causing the original Visual Studio 2010 linker to give the error "LNK1123: failure during conversion to COFF: file invalid or corrupt". The fix for this issue is to install Microsoft Visual Studio 2010 Service Pack 1* which can be found here.

3.5.5 Certain uses of length type parameters in parameterized derived types are not yet fully implemented

The following uses of length type parameters in parameterized derived types (PDTs) are not yet fully implemented:

- PDT parameter constants with length type parameters
- %RE and %IM are not yet implemented
- There is a syntax error in the FEE displaying extended types that are parameterized

3.5.6 /warn:interfaces may cause intermittent build fails with parallel builds

Using /warn:interfaces may cause intermittent build failures when performing parallel builds. When using /warn:interfaces, it is recommended to perform a serial build.

3.6 Microsoft Visual Studio 2010, 2012 and 2013 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++. These changes were carried forward into Visual Studio 2012/ 2013.

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_COMPILER15)\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_COMPILER15)\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 7 and Windows 8, it is typically:

```
C:\Users\<username>\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 2010/2012/2013. 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. (For Visual Studio 2013 select Build Dependencies > 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 2010/2012 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.

3.6.3 Showing Documentation Issue with Visual Studio 2012 and Windows Server* 2012 and Windows Server* 2008

If on Windows Server 2012* and Windows Server* 2008 you find that you cannot display help or documentation from within Visual Studio 2012, correcting a security setting for Microsoft Internet

Explorer* usually corrects the problem. From Tools > Internet Options > Security, change the settings for Internet Zone to allow “MIME Sniffing” and “Active Scripting”.

3.7 Fortran 2003 and Fortran 2008 Feature Summary

The Intel Fortran Compiler supports all features from the Fortran 2003 standard. The Intel® Fortran Compiler also supports many 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 ATOMIC_DEFINE, ATOMIC_REF, IMAGE_INDEX, LCOBOUND, NUM_IMAGES, THIS_IMAGE, UCOBOUND
- 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
- An OPTIONAL dummy argument that does not have the ALLOCATABLE or POINTER attribute, and which corresponds to an actual argument that: has the ALLOCATABLE attribute and is not allocated, or has the POINTER attribute and is disassociated, or is a reference to the NULL() intrinsic function, is considered not present
- A dummy argument that is a procedure pointer may be associated with an actual argument that is a valid target for the dummy pointer, or is a reference to the intrinsic

function NULL. If the actual argument is not a pointer, the dummy argument shall have the INTENT(IN) attribute.
- BLOCK construct
- intrinsic subroutine EXECUTE_COMMAND_LINE

4 Intel® Debugger Extension for Intel® Many Integrated Core Architecture (Intel® MIC Architecture)

This section summarizes new features and changes, usage and known issues related to the Intel® Debugger Extension. This debugger extension only supports code targeting Intel® Many Integrated Core Architecture (Intel® MIC Architecture).

4.1 Features

- Support for both native coprocessor applications and host applications with offload extensions

- Debug multiple coprocessor cards at the same time (with offload extension)

4.2 Using the Intel® Debugger Extension

The Intel® Debugger Extension is a plug-in for Microsoft Visual Studio* IDE. It transparently enables debugging of projects defined by that IDE. Applications for Intel® Xeon Phi™ coprocessors can be either loaded and executed or attached to.

Instructions on how to use Intel® Debugger Extension can be found in the documentation

4.3 Documentation

The full documentation for the Intel® Debugger Extension can be found here:

```
<install-dir>\Documentation\en_US|ja_JP]\debugger\ ↵
              mic\gdb_quickstart_win.pdf
```

4.4 Known Issues

- Offload debugging is only supported in Microsoft Visual Studio 2012* and Microsoft Visual Studio 2013*.
- Disassembly window cannot be scrolled outside of 1024 bytes from the starting address within an offload section.
- Handling of exceptions from the Intel® MIC Architecture application is not supported.
- Changing breakpoints while the application is running does not work. The changes will appear to be in effect but they are not applied.
- Starting an Intel® MIC Architecture native application is not supported. You can attach to a currently running application, though.
- The Thread Window in Microsoft Visual Studio* offers context menu actions to Freeze, Thaw and Rename threads. These context menu actions are not functional when the thread is on a coprocessor.
- Setting a breakpoint right before an offload section sets a breakpoint at the first statement of the offload section. This only is true if there is no statement for the host

between set breakpoint and offload section. This is normal Microsoft Visual Studio* breakpoint behavior but might become more visible with interweaved code from host and coprocessor. The superfluous breakpoint for the offload section can be manually disabled (or removed) if desired.

- Only Intel® 64 applications containing offload sections can be debugged with the Intel® Debugger Extension for Intel® Many Integrated Core Architecture.
- Stepping out of an offload section does not step back into the host code. It rather continues execution without stopping (unless another event occurs). This is intended behavior.
- The functionality “Set Next Statement” is not working within an offload section.
- If breakpoints have been set for an offload section in a project already, starting the debugger might show bound breakpoints without addresses. Those do not have an impact on functionality.
- For offload sections, using breakpoints with the following conditions of hit counts do not work: “break when the hit count is equal to” and “break when the hit count is a multiple of”.
- The following options in the Disassembly window do not work within offload sections: “Show Line Numbers”, “Show Symbol Names” and “Show Source Code”
- Evaluating variables declared outside the offload section shows wrong values.
- Please consult the Output (Debug) window for detailed reporting. It will name unimplemented features (see above) or provide additional information required to configuration problems in a debugging session. You can open the window in Microsoft Visual Studio* via menu *Debug->Windows->Output*.
- When debugging an offload enabled application, the debugger does not evaluate expressions that contain assignments which read memory locations before writing to them (e.g. x = x + 1). Please do not use such assignments when evaluating expressions (e.g. *Immediate Window*, *Watch* Window, …)
- Using conditional breakpoints for offload sections might stall the debugger. If a conditional breakpoint is created within an offload section, the debugger might hang when hitting it and evaluating the condition. This is currently analyzed and will be fixed with a future version of the product.
- Depending on the debugger extensions provided by Intel the behavior (e.g. run control) and output (e.g. disassembly) could differ to what is known from the Microsoft Visual Studio debugger. This is because of different debugging technologies used underneath. It is intended and does not have any disproportional impact on debugging experience.

5 Intel® Math Kernel Library (Intel® MKL)

This section summarizes changes, new features and late-breaking news about this version of the Intel® Math Kernel Library (Intel® MKL). Bug fixes can be found here.

5.1 What's New in Intel MKL 11.2 Update 4

- BLAS:

- Improved parallel and serial performance of ?TRSM on Intel® Advanced Vector Extensions 2 (Intel® AVX2) for 64-bit Intel MKL
- Improved ?SYRK/?HERK/?SYR2K/?HER2K performance for beta=0 on Intel® Advanced Vector Extensions 2 (Intel® AVX2) for 64-bit Intel MKL
- Improved serial performance of STRMM for small triangular matrices (dimension less than or equal to 10) on Intel® AVX2 for 64-bit Intel MKL
- Improved performance of BLAS level 3 functions for second generation of Intel® Xeon Phi™ coprocessors

5.2 What's New in Intel MKL 11.2 Update 3

- Extended the Intel MKL memory manager to improve scaling on large SMP systems.
- Added new service functions to provide more control for Intel MKL Automatic Offload for Intel Xeon Phi systems. These functions include mkl_mic_get_meminfo, mkl_mic_get_cpuinfo, mkl_mic_set_flags, mkl_mic_get_flags, mkl_mic_clear_status, and mkl_mic_get_status.
- BLAS:
  - Improved parallel performance of (D/S)SYMV on all Intel Xeon processors.
  - Improved (C/D/S/Z/DZ/SC)ROT performance for Intel Advanced Vector Extensions (Intel AVX) architectures in 64-bit Intel MKL.
  - Improved (C/Z)ROT performance for Intel Advanced Vector Extensions 2 (Intel AVX2) architectures in 64-bit Intel MKL.
  - Improved parallel performance of ?SYRK/?HERK, ?SYR2K/?HER2K, and ?GEMM for cases with large *k* sizes on Intel AVX2 architectures in 64-bit Intel MKL.
  - Improved ?SYRK/?HERK and ?SYR2K/?HER2K performance on Intel Xeon Phi coprocessors.
- LAPACK:
  - Improved performance of SVD, for cases where singular vectors are computed, on multi-socket systems based on Intel AVX or Intel AVX2 architectures.
  - Added new routines for incomplete LU factorization without pivoting.

5.3 What's New in Intel MKL 11.2 Update 2

- BLAS:
  - Improved ?GEMM performance for Intel® Xeon Phi™ coprocessors for cases where k >> m, k >> n.
  - Improved parallel and serial performance of ?HEMM/?SYMM for on Intel® Advanced Vector Extensions 2 (Intel® AVX2)for the 64-bit Intel MKL.
  - Improved parallel and serial performance of ?HERK/?SYRK and and ?HER2K/?SYR2K for Intel AVX2.

    - Added MKL_DIRECT_CALL support for CBLAS interfaces and ?GEMM3M routines.
    - Improved CGEMM performance for Intel® Advanced Vector Extensions 512 (Intel® AVX-512).
    - Improved SGEMM and ZGEMM performance for AMD* Opteron* 6000 series.
    - Small performance improvement for CGEMM and ZGEMM for Intel AVX2 for the 64-bit Intel MKL.
- LAPACK:
    - Improved symmetric eigensolvers performance by up to 3x, for the cases when eigenvectors are not needed.
    - Improved ?GESVD performance by 2-3x, for the cases when singular vectors are required.
    - Improved ?GETRF performance for Intel AVX2 by up to 14x for non-square matrices.
    - Narrowed the ?GETRF performance gap between CNR (Conditional Numerical Reproducibility)-enabled and CNR-disabled cases. The gap is now below 5%.
    - Improved Intel® Optimized LINPACK Benchmark shared memory (SMP) implementation performance for Intel AVX2 by up to 40%.
- Parallel Direct Sparse Solver for Clusters:
    - Added ability to overwrite the right hand side vector with solution with the distributed CSR format.
    - Added ability to gather system solution on all compute nodes with distributed CSR format.
- Intel® MKL PARDISO:
    - Significantly improved overall scalability for Intel Xeon Phi coprocessors.
    - Improved the scalability of the solving step for Intel® Xeon® processors.
    - Reduced memory footprint in the out-of-core mode.
    - Added ability to free up memory used by the input matrix after the factorization step. This helps to reduce memory consumption when iterative refinement is not needed and disabled by the user.
- Extended Eigensolver:
    - Improved performance for Intel Xeon processors
- VSL:
    - Summary Statistics:
        - Improved performance of variance-covariance matrices computation and correlation matrices computation routines for cases when the task dimension is approximately equal to the number of observations.
    - RNG:

        - Improved performance of the Sobol and the Niederreiter Quasi-RNGs for Intel Xeon processors.
- Convolution and correlation:
    - Improved 3D convolution performance.

5.4 What's New in Intel MKL 11.2 Update 1

- Introduced support for Intel® Advanced Vector Extensions 512 (Intel® AVX-512) on Intel® Xeon® processors for Windows* and Linux* versions of Intel MKL. This is in addition to the current support for Intel® AVX-512 instructions for Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
- BLAS:
    - Optimized the following functions on Intel microarchitecture code name Skylake:
        - (D/Z)AXPY,(S/D/C/Z)COPY, DTRMM (for cases when the triangular matrix is on right side and with no matrix transpose)
    - Optimized following BLAS Level-1 functions on Intel AVX2 both for Intel® 64 and IA-32 architectures
        - (S/D)DOT,(S/D)SCAL,(S/D)ROT,(S/D)ROTM,(S/D/C/Z)SWAP,(S/D/SC/DZ)ASUM
    - Improved ?GEMM performance (serial and multithreaded) on Intel AVX2 (for IA-32 architectures)
    - Improved ?GEMM performance for beta==0 on Intel AVX and Intel AVX2 (for Intel 64 architectures)
    - Improved DGEMM performance (serial and multithreaded) on Intel AVX (for Intel 64 architectures)
- LAPACK:
    - Introduced support for LAPACK version 3.5. New features introduced in this version are:
        - Symmetric/Hermitian LDLT factorization routines with rook pivoting algorithm
        - 2-by-1 CSD for tall and skinny matrix with orthonormal columns
    - Improved performance of (C/Z)GE(SVD/SDD) when M>=N and singular vectors are not needed
- FFT:
    - Introduced Automatic Offload mode for 1D Batch FFT on Intel MIC Architecture

    - Improved performance of Hybrid (OpenMP+MPI) Cluster FFT
    - Improved accuracy of large 1D real-to-complex transforms
- Parallel Direct Sparse Solver for Clusters:
    - Added support for many factorization steps with the same reordering (maxfct > 1)
- Intel MKL PARDISO:
    - Added support for Schur complement, including getting explicit Schur complement matrix and solving the system through Schur complement
- Sparse BLAS:
    - Optimized SpMV on Intel microarchitecture code name Skylake
    - Added Sparse Matrix Checker functionality as standalone API to simplify validation of matrix structure and indices(see Sparse Matrix Checker Routines in Intel® Math Kernel Library (Intel® MKL) Reference Manual
    - Sparse BLAS API for C/C++ uses const modifier for constant parameters
- VML:
    - Introduced new Environment variable, MKL_VML_MODE to control the accuracy behavior. This Environment variable can be used to control VML functions behavior (analog of vmlSetMode() function)

5.5 What's New in Intel MKL 11.2

- Intel MKL now provides optimizations for all Intel® Atom™ processors that support Intel® Streaming SIMD Extensions 4.1 (Intel® SSE4.1) and Intel® Streaming SIMD Extensions 4.2 (Intel® SSE4.2) instruction sets
- Introduced support for Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instruction set with limited optimizations in BLAS, DFT and VML
- Introduced Verbose support for BLAS and LAPACK domains, which enables users to capture the input parameters to Intel MKL function calls
- Introduced support for Intel® MPI Library 5.0
- Introduced the Intel Math Kernel Library Cookbook (http://software.intel.com/en-us/mkl_cookbook) , a new document that describes how to use Intel MKL routines to solve certain complex problems
- Introduced the MKL_DIRECT_CALL or MKL_DIRECT_CALL_SEQ compilation feature that provides ?GEMM small matrix performance improvements for all processors (see the *Intel® Math Kernel Library User's Guide* for more details)
- Added the ability to link a Single Dynamic Library (mkl_rt) on Intel® Many Integrated Core Architecture (Intel® MIC Architecture)

- Added a customizable error handler. See the *Intel Math Kernel Library Reference Manual* description of mkl_set_exit_handler() for further details
- Extended the Intel® Xeon Phi™ coprocessor Automatic Offload feature with a resource sharing mechanism.See the *Intel Math Kernel Library Reference Manual* for the description of mkl_mic_set_resource_limit() function and the MKL_MIC_RESOURCE_LIMIT environment variable for further details
- Parallel Direct Sparse Solver for Clusters:
    - Introduced Parallel Direct Sparse Solver for Clusters, a distributed memory version of Intel MKL PARDISO direct sparse solver
    - Improved performance of the matrix gather step for distributed matrices
    - Enabled reuse of reordering information on multiple factorization steps
    - Added distributed CSR format, support of distributed matrices, RHS, and distributed solutions
    - Added support of solving of systems with multiple right hand sides
    - Added cluster support of factorization and solving steps
    - Added support for pure MPI mode and support for single OpenMP thread in hybrid configurations
- BLAS:
    - Improved threaded performance of ?GEMM for all 64-bit architectures supporting Intel® Advanced Vector Extensions 2 (Intel® AVX2)
    - Optimized ?GEMM, ?TRSM, DTRMM for the Intel AVX-512 instruction set
    - Improved performance of ?GEMM for outer product [large m, large n, small k] and tall skinny matrices [large m, medium n, small k] on Intel MIC Architecture
    - Improved performance of ?TRSM and ?SYMM in Automatic Offload mode on Intel MIC Architecture
    - Improved performance of Level 3 BLAS functions for 64-bit processors supporting Intel AVX2
    - Improved ?GEMM performance on small matrices for all processors when MKL_DIRECT_CALL or MKL_DIRECT_CALL_SEQ is defined during compilation (see the *Intel® Math Kernel Library User's Guide for more details*)
    - Improved performance of DGER and DGEMM for the beta=1, k=1 case for 64-bit processors supporting Intel SSE4.2, Intel® Advanced Vector Extensions (Intel® AVX), and Intel AVX2 instruction sets
    - Optimized (D/Z)AXPY for the Intel AVX-512 instruction set
    - Optimized ?COPY for Intel AVX2 and AVX-512 instruction sets
    - Optimized DGEMV for Intel AVX-512 instruction set
    - Improved performance of SSYR2K for 64-bit processors supporting Intel AVX and Intel AVX2
    - Improved threaded performance of ?AXPBY for all Intel processors

    - Improved DTRMM performance for the side=R, uplo={U,L}, transa=N, diag={N,U} cases for Intel AVX-512
- LINPACK:
    - Improved performance of matrix generation in the heterogeneous Intel® Optimized MP LINPACK Benchmark for Clusters
    - Intel MIC Architecture offload option of the Intel Optimized MP LINPACK Benchmark for Clusters package now supports Intel AVX2 hosts
    - Improved performance of the Intel Optimized MP LINPACK for Clusters package for 64-bit processors supporting Intel AVX2
- LAPACK:
    - Improved performance of ?(SY/HE)RDB
    - Improved performance of ?(SY/HE)EV when eigenvectors are needed
    - Improved performance of ?(SY/HE)(EV/EVR/EVD) when eigenvectors are not needed
    - Improved performance of ?GELQF,?GELS and ?GELSS for underdetermined case (M less than N)
    - Improved performance of ?GEHRD,?GEEV and ?GEES
    - Improved performance of NaN checkers in LAPACKE interfaces
    - Improved performance of ?GELSX, ?GGSVP
    - Improved performance of ?(SY/HE)(EV/EVR/EVD) when eigenvectors are not needed
    - Improved performance of ?GETRF
    - Improved performance of (S/D)GE(SVD/SDD) when M>=N and singular vectors are not needed
    - Improved performance of ?POTRF UPLO=U in Automatic Offload mode on Intel MIC Architecture
    - Added Automatic Offload for ?SYRDB on Intel MIC Architecture, which speeds up ?SY(EV/EVD/EVR) when eigenvectors are not needed
- PBLAS and ScaLAPACK:
    - Enabled Automatic Offload in P?GEMM routines for large distribution blocking factors
- Sparse BLAS:
    - Optimized SpMV kernels for Intel AVX-512 instruction set
    - Added release example for diagonal format use in Sparse BLAS
    - Improved Sparse BLAS level 2 and 3 performance for systems supporting Intel SSE4.2, Intel AVX and Intel AVX2 instruction sets
- Intel MKL PARDISO:
    - Added the ability to store Intel MKL PARDISO handle to the disk for future use at any solver stage

    - Added pivot control support for unsymmetric matrices and out-of-core mode
    - Added diagonal extraction support for unsymmetric matrices and out-of-core mode
    - Added example demonstrating use of Intel MKL PARDISO as iterative solver for non-linear systems
    - Added capability to free memory taken by original matrix after factorization stage if iterative refinement is disabled
    - Improved memory estimation of out-of-core (OOC) portion size for reordering algorithm leading to improved factorization-solve performance in OOC mode
    - Improved message output from Intel MKL PARDISO
    - Added support of zero pivot during factorization for structurally symmetric cases
- Poisson library:
    - Added example demonstrating use of the Intel MKL Poisson library as a preconditioner for linear systems solves
- Extended Eigensolver:
    - Improved message output
    - Improved examples
    - Added input and output iparm parameters in predefined interfaces for solving sparse problems
- FFT:
    - Optimized FFTs for the Intel AVX-512 instruction set
    - Improved performance for non-power-of-2 length on Intel® MIC Architecture
- VML: Added v[d|s]Frac function computing fractional part for each vector element
- VSL RNG:
    - Added support of ntrial=0 in Binomial Random Number Generator
    - Improved performance of MRG32K3A and MT2203 BRNGs on Intel MIC Architecture
    - Improved performance of MT2203 BRNG on CPUs supporting Intel AVX and Intel AVX2 instruction sets
- VSL Summary Statistics:
    - Added support for group/pooled mean estimates (VSL_SS_GROUP_MEAN/VSL_SS_POOLED_MEAN)
- Data Fitting: Fixed incorrect behavior of the natural cubic spline construction function when number of breakpoints is 2 or 3
- Introduced an Intel MKL mode that ignores all settings specified by Intel MKL environment variables
    - User can set up the mode by calling mkl_set_env_mode() routine which directs Intel MKL to ignore all environment settings specific to Intel MKL so that all Intel MKL related environment variables such as MKL_NUM_THREADS,

MKL_DYNAMIC, MKL_MIC_ENABLE and others are ignored; users can instead set needed parameters via Intel MKL service routines such as mkl_set_num_threads() and mkl_mic_enable()

5.6 Notes

- Intel MKL now provides a choice of components to install. Components necessary for PGI compiler, Compaq Visual Fortran Compiler, SP2DP interface, BLAS95 and LAPACK95 interfaces, Cluster support (ScaLAPACK and Cluster DFT) and Intel MIC Architecture support are not installed unless explicitly selected during installation
- Unaligned CNR is not available for MKL Cluster components (ScaLAPACK and Cluster DFT)
- Examples for using Intel MKL with BOOST/uBLAS and Java have been removed from the product distribution and placed in the following articles:
  - How to use Intel® MKL with Java*
  - How to use BOOST* uBLAS with Intel® MKL
- API symbols, order of arguments and link line have changed since Intel MKL 11.2 Beta Update 2. (see the *Intel® Math Kernel Library User's Guide* for more details)
- Important deprecations are listed in Intel® Math Kernel Library (Intel® MKL) 11.2 Deprecations

5.7 Known Issues

- Automatic Offload on Windows with large matrices may cause data corruption or crash. There is a problem in COI: HSD4868293 (critical). COI Cannot allocate a buffer with >= 2**32 bytes and 2M pages on Windows

  Workaround: Set MKL_MIC_MAX_MEMORY=3G. Note: This issue is resolved in Intel® MPSS 3.3

A full list of the known limitations can be found in the Intel® MKL Article List at Intel® Developer Zone

5.8 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. 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.

The Intel® MKL Extended Eigensolver functionality is based on the Feast Eigenvalue Solver 2.0 (http://www.ecs.umass.edu/~polizzi/feast/)

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:

http://www.intel.com/products/processor_number/

for details.

The Visual Intel® Fortran Compiler and Intel® Math Kernel Library are provided under Intel Corporation's End User License Agreement (EULA).

The GNU* Project Debugger, GDB is provided under the General GNU Public License, GPL V3.

Celeron, Centrino, Intel, Intel logo, Intel386, Intel486, Intel Atom, Intel Core, Itanium, MMX, Pentium, VTune, Xeon and Xeon Phi 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 © 2015 Intel Corporation. All Rights Reserved.