

# Intel® Fortran Composer XE 2013 for Linux\*

## Installation Guide and Release Notes

---

Document number: 321415-004US

24 January 2013

### Table of Contents

1	Introduction .....	4
1.1	Change History .....	4
1.1.1	Product Updates .....	4
1.1.2	Changes since Intel® Fortran Composer XE 2011 .....	5
1.2	Product Contents .....	5
1.3	System Requirements.....	5
1.4	Documentation.....	7
1.5	Optimization Notice.....	7
1.6	Japanese Language Support.....	7
1.7	Technical Support.....	8
2	Installation.....	8
2.1	Cluster Installation .....	8
2.2	Installation of Intel® Manycore Platform Software Stack (Intel® MPSS) .....	9
2.3	Intel® Software Manager .....	9
2.4	Silent Install .....	9
2.5	Using a License Server .....	9
2.6	Known Installation Issues.....	9
2.7	Installation Folders.....	10
2.8	Removal/Uninstall .....	11
3	Intel® Fortran Compiler.....	12
3.1	Compatibility .....	12
3.1.1	Stack Alignment Change for REAL(16) and COMPLEX(16) Datatypes.....	12
3.2	New and Changed Features .....	12

3.2.1	Features from Fortran 2003 .....	12
3.2.2	Features from Fortran 2008 .....	13
3.2.3	Intel® Many Integrated Core (Intel® MIC) .....	13
3.2.4	Naming of compiled module files when using OFFLOAD .....	13
3.2.5	New and Changed Directives .....	13
3.2.6	OpenMP Changes .....	13
3.2.7	ATTRIBUTES ALIGN for component of derived type (13.0.1) .....	15
3.2.8	Additional Compiler Changes .....	15
3.2.9	Change in File Buffering Behavior (13.1) .....	15
3.3	New and Changed Compiler Options .....	16
3.3.1	New -fimf-domain-exclusion Compiler Option .....	17
3.3.2	New -vec-report7 Compiler Option (13.1.0) .....	18
3.4	Other Changes and Notes .....	18
3.4.1	Establishing the Compiler Environment .....	18
3.5	Known Issues .....	18
3.5.1	Coarray Issues .....	18
3.6	Coarrays .....	19
3.6.1	How to Debug a Coarray Application .....	19
3.6.2	Compiler Option to Improve Coarray Performance (13.0.1) .....	21
3.6.3	Coarray Known Issues .....	21
3.7	Fortran 2003 and Fortran 2008 Feature Summary .....	21
4	Intel® Debugger (IDB) .....	25
4.1	Support Deprecated for Intel® Debugger .....	25
4.2	Setting up the Java* Runtime Environment .....	25
4.3	Starting the Debugger .....	25
4.4	Additional Documentation .....	26
4.5	Debugger Features .....	26
4.6	Known Issues and Changes .....	26
4.6.1	Coarray elements cannot be viewed. ....	26
4.6.2	Signals Dialog not working Signals Dialog not working .....	26
4.6.3	Resizing GUI .....	26
4.6.4	\$cdir, \$cwd Directories .....	26

4.6.5	info stack Usage .....	26
4.6.6	\$stepg0 Default Value Changed .....	27
4.6.7	SIGTRAP error on some Linux* Systems.....	27
4.6.8	idb GUI cannot be used to debug MPI processes .....	27
4.6.9	Thread Syncpoint Creation in GUI .....	27
4.6.10	Stack Alignment for IA-32 Architecture.....	27
4.6.11	GNOME Environment Issues .....	27
4.6.12	Accessing Online-Help.....	28
4.6.13	Debugger crashes if \$HOME not set on calling shell.....	28
4.6.14	Command line parameter –parallel not supported.....	28
4.6.15	Command line parameter –idb and -dbx not supported.....	28
4.6.16	Core File Debugging .....	28
5	Intel® Xeon Phi™ Coprocessors.....	28
5.1	Introduction .....	28
5.2	Documentation.....	29
5.3	Debugger.....	29
5.4	Changes and Known Issues .....	29
5.4.1	*MIC* tag added to compile-time diagnostics .....	29
5.4.2	Direct (native) mode requires transferring libiomp5.so to coprocessor .....	29
5.4.3	Tuning Memory Allocation Performance .....	30
5.4.4	Stepping “A” Hardware Requires –opt-streaming-stores never .....	30
5.4.5	Runtime errors or crashes when running an application built with the initial Intel® Composer XE 2013 product release with the offload libraries from a later update.....	30
5.4.6	Non-Contiguous Array Sections May Not Be Passed to an Offload Region .....	31
5.4.7	Environment Variable for Controlling Offload Behavior .....	31
5.4.8	OFFLOAD_DEVICES .....	33
5.4.9	Debugging and Intel® Debugger.....	33
6	Intel® Math Kernel Library .....	34
6.1	What’s New in Intel® MKL 11.0 Update 2 .....	34
6.2	What’s New in Intel® MKL 11.0 Update 1 .....	35
6.3	What’s New in Intel® MKL 11.0 .....	36
6.4	Deprecated and Removed Features .....	38

6.5	Known Issues .....	38
6.6	Attributions.....	38
7	Disclaimer and Legal Information.....	39

## 1 Introduction

This document describes how to install the product, provide 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 from the previous product version and changes in product updates. For information on what is new in each component, please read the individual component release notes.

#### 1.1.1 Product Updates

##### Update 2 – January 2013

- Intel® Fortran Compiler [updated to 13.1.0](#)
  - Added support for additional directives, clauses and procedures from [OpenMP\\* 4.0](#)
  - Added the [KMP\\_PLACE\\_THREADS run-time environment variable](#)
  - Added (in Update 1) a temporary [option that can improve performance of coarray applications](#)
  - Added [–vec-report7 compiler option](#)
  - Added description of [change in how sequential, unformatted files are buffered during READs](#)
  - Added description of how [compiled module files are named when offload regions are used](#)
  - Added note on [restriction on non-contiguous array sections passed into an offload region](#)
- Intel® Math Kernel Library [updated to 11.0 Update 2](#)
- Corrections to reported problems
  - Compiler fix list: <http://intel.ly/S5uIAb>
  - Intel® MKL fix list: <http://intel.ly/S5uw3R>

##### Update 1 – October 2012

- Intel® Fortran Compiler [updated to 13.0.1](#)
  - [ATTRIBUTES ALIGN may now be specified](#) for an ALLOCATABLE or POINTER component of a derived type
- Intel® Math Kernel Library [updated to 11.0 Update 1](#)

- Documentation on [\\_fimf-domain-exclusion](#) added
- [Breaking binary compatibility change in Update 1 offload libraries for Intel® Many Integrated Core Architecture](#)
- [-opt-streaming-stores never option must be used](#) when compiling for Stepping “A” of Intel® Xeon Phi™ coprocessors
- Corrections to reported problems
  - Compiler fix list: <http://intel.ly/S5uIAb>
  - Intel® MKL fix list: <http://intel.ly/S5uw3R>

### 1.1.2 Changes since Intel® Fortran Composer XE 2011

- Development of applications that offload work to or natively run on an Intel® Many Integrated Core (Intel® MIC) architecture coprocessor (Intel® Xeon Phi™ product family) is now supported. For details, see the section on [Intel® Xeon Phi™ Coprocessors](#)
- Intel® Fortran Compiler [updated to version 13.0](#)
- Intel® Debugger [updated to version 13.0](#)
  - [Intel® Debugger support deprecated](#)
- Intel® Math Kernel Library [updated to version 11.0](#)
  - Intel® Math Kernel Library removed support for Intel® Pentium® III processors. The minimum instruction set supported by Intel® MKL is Intel® SSE2. See [Intel® MKL section](#) for additional changes
- Support for SUSE\* LINUX Enterprise Server 11 SP2 added. Support for SP1 dropped.
- Support for Fedora\* 17, Ubuntu 12.04 and Ubuntu 11.10\* added.
- Support for the following versions of Linux distributions has been dropped:
  - Asianux\*
  - Red Hat Enterprise Linux 4\*
  - Fedora 15\*
  - Ubuntu 11.04\*
- The Intel® Software Manager [has been added](#) to help you manage product updates and license activation
- Corrections to reported problems

## 1.2 Product Contents

*Intel® Fortran Composer XE 2013 for Linux\** includes the following components:

- Intel® Fortran Compiler XE 13.1.0 for building applications that run on IA-32, Intel® 64 architecture systems and Intel® Xeon Phi™ coprocessors running the Linux\* operating system
- Intel® Debugger 13.1.0
- Intel® Math Kernel Library 11.0 Update 2
- On-disk documentation

## 1.3 System Requirements

For an explanation of architecture names, see <http://intel.ly/mXlIjK>

Intel® Fortran Composer XE 2013 for Linux\* Installation Guide and Release Notes

- A PC based on an IA-32 or Intel® 64 architecture processor supporting the Intel® Streaming SIMD Extensions 2 (Intel® SSE2) instructions (Intel® Pentium® 4 processor or later, or compatible non-Intel processor)
  - Development of 64-bit applications, and those that offload work to Intel® Xeon Phi™ coprocessors, is supported on a 64-bit version of the OS only. Development of 32-bit applications is supported on either 32-bit or 64-bit versions of the OS.
  - Development for a 32-bit on a 64-bit host may require optional library components (ia32-libs, lib32gcc1, lib32stdc++6, libc6-dev-i386, gcc-multilib) to be installed from your Linux distribution.
- For the best experience, a multi-core or multi-processor system is recommended
- 1GB of RAM (2GB recommended)
- 2.5GB free disk space for all features
- For Intel® Xeon Phi™ coprocessor development/testing:
  - Intel® Manycore Platform Software Stack (Intel® MPSS)
- For development of IA-32 or Intel® 64 architecture applications, one of the following Linux distributions (this is the list of distributions tested by Intel; other distributions may or may not work and are not recommended - please refer to [Technical Support](#) if you have questions):
  - Debian\* 6.0
  - Fedora\* 17
  - Red Hat Enterprise Linux\* 5, 6
  - SuSE LINUX Enterprise Server\* 10,11 SP2
  - Ubuntu\* 11.10, 12.04
  - Intel® Cluster Ready
  - Pardus\* 2011.2 (x64 only)
- Linux Developer tools component installed, including gcc, g++ and related tools
- Library `libunwind.so` is required in order to use the `-traceback` option. Some Linux distributions may require that it be obtained and installed separately.

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

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

### **Notes**

- The Intel® compilers are tested with a number of different Linux distributions, with different versions of gcc. Some Linux distributions may contain header files different from those we have tested, which may cause problems. The version of glibc you use must be consistent with the version of gcc in use. For best results, use only the gcc versions as supplied with distributions listed above.

- The default for the Intel® compilers is to build IA-32 architecture applications that require a processor supporting the Intel® SSE2 instructions - for example, the Intel® Pentium® 4 processor. A compiler option is available to generate code that will run on any IA-32 architecture processor. However, Intel® MKL requires Intel® SSE2 as a minimum instruction set.
- Compiling very large source files (several thousands of lines) using advanced optimizations such as -O3, -ipo and -openmp, may require substantially larger amounts of RAM.
- Some optimization options have restrictions regarding the processor type on which the application is run. Please see the documentation of these options for more information.

## 1.4 Documentation

Product documentation can be found in the `Documentation` folder as shown under [Installation Folders](#).

## 1.5 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

## 1.6 Japanese Language Support

Intel® compilers optionally provide support for Japanese language users when the combined English-Japanese product is installed. 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 is not provided with every update of the product.

If you wish to use Japanese-language support on an English-language operating system, or English-language support on a Japanese-language operating system, you will find instructions at <http://intel.ly/qhINDv>

Intel® Fortran Composer XE 2013 for Linux\* Installation Guide and Release Notes

## 1.7 Technical Support

Register your license at the [Intel® Software Development Products Registration Center](http://www.intel.com/software/products/support/). 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

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, mount the DVD, change the directory (`cd`) to the top-level directory of the mounted DVD and begin the installation using the command:

```
./install.sh
```

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

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

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

```
./install.sh
```

Follow the prompts to complete installation.

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.

### 2.1 Cluster Installation

If a license for Intel® Cluster Studio XE is present, and the installation detects that the installing system is a member of a cluster, you will have the option of installing on multiple nodes of the cluster.



To install on multiple nodes, follow these steps:

1. Passwordless ssh must be configured among the nodes of the cluster
2. During install step 4, “Options”, select “Cluster installation”.
3. You will be prompted to provide the path to a `machines.LINUX` file with IP addresses, hostnames, or Fully Qualified Domain Names (FQDNs) of the cluster nodes, one per line. The first line is expected to be the current (master) node.
4. Once the `machines.LINUX` file is found, additional options will appear, including “Number of parallel installations” and “Check for shared installation directory”. Select the desired options.
5. Once all options are configured and the install is started, the installation will check connectivity to all the nodes; if successful, it will attempt the install on all indicated nodes.

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

The Intel® Manycore Platform System Software (Intel® MPSS) may be installed before or after installing the Intel® Composer XE 2013 for Linux\* product.

Refer to the Intel® MPSS documentation for the necessary steps to install the user space and kernel drivers.

## 2.3 Intel® Software Manager

The installation now provides an 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 <http://intel.ly/SoftwareImprovementProgram>

## 2.4 Silent Install

For information on automated or “silent” install capability, please see <http://intel.ly/ngVHY8>

## 2.5 Using a License Server

If you have purchased a “floating” license, see <http://intel.ly/oPEdEe> for information on how to install using a license file or license server. 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.

## 2.6 Known Installation Issues

- On some versions of Linux, auto-mounted devices do not have the “exec” permission and therefore running the installation script directly from the DVD will result in an error such as:

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

Intel® Fortran Composer XE 2013 for Linux\* Installation Guide and Release Notes

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

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

and then try the installation again.

- The product is fully supported on Ubuntu and Debian Linux distributions for IA-32 and Intel® 64 architecture systems as noted above under System Requirements. Due to a restriction in the licensing software, however, it is not possible to use the Trial License feature when evaluating IA-32 components on an Intel® 64 architecture system under Ubuntu or Debian. This affects using a Trial License only. Use of serial numbers, license files, floating licenses or other license manager operations, and off-line activation (with serial numbers) is not affected. If you need to evaluate IA-32 components of the product on an Intel® 64 architecture Ubuntu or Debian system, please visit the Intel® Software Evaluation Center (<http://intel.ly/nJS8y8>) to obtain an evaluation serial number.

## 2.7 Installation Folders

The compiler installs, by default, under `/opt/intel` – this is referenced as `<install-dir>` in the remainder of this document. You are able to specify a different location, and can also perform a “non-root” install in the location of your choice.

Under `<install-dir>` are the following directories:

- `bin` – contains symbolic links to executables for the latest installed version
- `lib` – symbolic link to the `lib` directory for the latest installed version
- `include` – symbolic link to the `include` directory for the latest installed version
- `man` – symbolic link to the directory containing man pages for the latest installed version
- `mkl` – symbolic link to the directory for the latest installed version of Intel® Math Kernel Library
- `composerxe` – symbolic link to the `composer_xe_2013` directory
- `composer_xe_2013` – directory containing symbolic links to subdirectories for the latest installed Intel® Composer XE 2013 product release
- `composer_xe_2013.<n>.<pkg>` - physical directory containing files for a specific update version. `<n>` is the update number, and `<pkg>` is a package build identifier

Each `composer_xe_2013` directory contains the following directories that reference the latest installed Intel® Composer XE 2013 product:

- `bin` – directory containing scripts to establish the compiler environment and symbolic links to compiler executables for the host platform
- `pkg_bin` – symbolic link to the compiler `bin` directory
- `include` – symbolic link to the compiler `include` directory
- `lib` – symbolic link to the compiler `lib` directory
- `mkl` – symbolic link to the `mkl` directory

Intel® Fortran Composer XE 2013 for Linux\* Installation Guide and Release Notes

- `debugger` – symbolic link to the `debugger` directory
- `man` – symbolic link to the directory containing man pages for the latest installed version
- `Documentation` – symbolic link to the `documentation` directory
- `Samples` – symbolic link to the `samples` directory
- `eclipse_support` – symbolic link to a directory created by the Intel Debugger component that is shared between Intel Fortran and Intel C++. Intel does not provide Eclipse support for Fortran.

Each `composer_xe_2013.<n>.<pkg>` directory contains the following directories that reference a specific update of the Intel® Composer XE 2013 compiler:

- `bin` – all executables
- `compiler` – shared libraries and include/header files
- `debugger` – debugger files
- `Documentation` – documentation files
- `eclipse_support` – directory created by the Intel Debugger component that is shared between Intel Fortran and Intel C++. Intel does not provide Eclipse support for Fortran.
- `man` – man pages
- `mkl` – Intel® Math Kernel Library libraries and header files
- `mpirt` – Intel® MPI Library run-time files used by Fortran coarray support
- `Samples` – Product samples and tutorial files

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

This directory layout allows you to choose whether you want the latest product update, no matter which version, the latest update of the Intel® Composer XE 2013 product, or a specific update. Most users will reference `<install-dir>/bin` for the `compilervars.sh` [`.csh`] script, which will always get the latest product installed. This layout should remain stable for future releases.

## 2.8 Removal/Uninstall

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

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

If you also have the same-numbered version of Intel® C++ Compiler installed, it may also be removed.

## 3 Intel® Fortran Compiler

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

### 3.1 Compatibility

In general, object code and modules compiled with earlier versions of Intel Fortran Compiler for Linux\* (8.0 and later) may be used in a build with version 13. 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 (-ipo) 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 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

In versions prior to 12.0, 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 and later compilers align such items at 16 bytes and expects received arguments to be aligned on 16-byte boundaries. This change is also compatible with gcc.

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 if they were compiled by compiler versions earlier than 12.0.

### 3.2 New and Changed Features

#### 3.2.1 Features from Fortran 2003

- Default initialization of polymorphic variables
- The keyword MODULE may be omitted from MODULE PROCEDURE in a generic interface block when referring to an external procedure

### 3.2.2 Features from Fortran 2008

- ATOMIC\_DEFINE and ATOMIC\_REF

### 3.2.3 Intel® Many Integrated Core (Intel® MIC)

Support has been added to build applications that offload work to an Intel® Many Integrated Core (Intel® MIC) architecture coprocessor (Intel® Xeon Phi™ product family). This support adds the following features:

- ATTRIBUTES OFFLOAD directive
- OFFLOAD and END OFFLOAD directives
- OMP OFFLOAD directive
- Intrinsic functions OFFLOAD\_GET\_DEVICE\_NUMBER, OFFLOAD\_NUMBER\_OF\_DEVICES
- Sample programs under samples/mic\_samples

For more information on how to write, build and run applications that use Intel® MIC, please see [Intel® Xeon Phi™ Coprocessors](#) and the compiler documentation

### 3.2.4 Naming of compiled module files when using OFFLOAD

When a Fortran source that uses the OFFLOAD directive, to offload work to an Intel® Xeon Phi™ coprocessor, is compiled, the source is compiled twice – once for native execution and once for execution on the coprocessor. If the source contains module subprograms, these are compiled twice as well. If the module source has dependencies on the execution target, this can cause the compiled module files to be different. If the same file naming convention was used, the compilation for Intel® MIC architecture would overwrite the compilation for native code.

The Intel® Fortran compiler resolves this problem by creating the compiled module file for Intel® MIC architecture with a .modmic file type rather than the default .mod file type, but this is done if and only if the module source includes an offload region. When compiling for Intel® MIC architecture, the compiler first looks for a .modmic file – if that does not exist then it looks for a .mod file. You may need to adjust makefiles or build scripts to accommodate this different behavior.

### 3.2.5 New and Changed Directives

The following general compiler directives are new or changed in Intel® Composer XE 2013 – please see the documentation for details:

- ATTRIBUTES CVF
- ATTRIBUTES OFFLOAD
- OFFLOAD/END OFFLOAD
- ORDERED/END ORDERED
- SIMD VECTORLENGTHFOR

### 3.2.6 OpenMP Changes

The following changes to OpenMP\* support are in Intel® Composer XE 2013

Intel® Fortran Composer XE 2013 for Linux\* Installation Guide and Release Notes

- OMP OFFLOAD directive

### **3.2.6.1 OpenMP 4.0 Changes (13.1.0)**

The following directives, clauses and procedures, new in OpenMP\* 4.0, are supported by the 13.1.0 (Composer XE 2013 Update 2) compiler. Until the product documentation is updated, please refer to <http://intel.ly/VPV24X>

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
- OMP DECLARE TARGET MIRROR
- OMP DECLARE TARGET LINKABLE

Clauses:

- MAP
- MAPFROM
- MAPTO
- SCRATCH

Procedures:

- OMP\_GET\_DEVICE\_NUM
- OMP\_GET\_PROC\_BIND
- OMP\_SET\_DEVICE\_NUM

### **3.2.6.2 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.

#### **Syntax**

```
value = ( int [ "C" | "T" ] [ delim ] | delim ) [ int [ "T" ]  
[ delim ] ] [ int [ "O" ] ];
```

```
delim = ", " | "x";
```

## Effect

Specifies the number of cores, with optional offset value and number of threads per core to use. The "C" indicates cores, "T" indicates threads, "O" is used to specify an offset. Either cores or threads should be specified. If omitted, the default value is the available number of cores (threads).

## Examples

```
5C,3T,1O - use 5 cores with offset 1, 3 threads per core
5,3,1    - same as above
24       - use first 24 cores, all available threads per core
2T       - use all cores, 2 threads per core
,2       - same as above
3x2      - use 3 cores, 2 threads per core
4C12O    - use 4 cores with offset 12, all available threads per core
```

### 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 Additional Compiler Changes

- The output of the G format edit descriptor has been changed to more properly conform to the Fortran 2008 standard. The changes involve effects of rounding on selected format, representation of values that round to -0
- When on output using a D, E, G, EN or ES format the exponent field overflows the implicit exponent width, the output field is filled with asterisks. In earlier versions, the exponent would be displayed in a manner inconsistent with the standard
- The compiler can now vectorize references to the RANDOM\_NUMBER and RANF intrinsic subroutines.

### 3.2.9 Change in File Buffering Behavior (13.1)

In product versions prior to Intel® 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® 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® 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.

### 3.3 New and Changed Compiler Options

Please refer to the compiler documentation for details

- `-align array8byte`
- `-align array16byte`
- `-align array32byte`
- `-align array64byte`
- `-align array128byte`
- `-align array256byte`
- `-assume [no]std_intent_in`
- `-diag-enable sc-enums`
- `-diag-enable sc-{full|concise|precise}`
- `-diag-enable sc-single-file`



- -fimf-domain-exclusion=classlist[:funclist] (See [below](#).)
- -guide-profile=<file|dir>[, [file|dir], ...]
- -mmic
- -no-offload
- -offload-attribute-target=<name>
- -offload-option<target>,<tool>,"option list"
- -openmp-link <library>
- -vec-report6
- [-vec-report7](#) (13.1.0)

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

### 3.3.1 New -fimf-domain-exclusion Compiler Option

The following information was omitted from the on-disk documentation.

#### **fimf-domain-exclusion**

Indicates the domain on which a function is evaluated.

#### **IDE Equivalent**

None

#### **Architectures**

Intel® 64 architecture, targeting Intel® MIC Architecture

#### **Syntax**

```
-fimf-domain-exclusion=classList[:funcList]
```

#### **Arguments**

*classList*            A comma separated list of:

- One or more of the five floating point value class names you can exclude from the function domain without affecting the correctness of your program
- One of the short-hand tokens.

The class names are:

- extremes
- nans
- infinities
- denormals
- zeros

The short-hand tokens are:

- none: None of the above are excluded from the domain.
- all: All of the above are excluded from the domain.
- common: Same as extremes,nans,infinities,denormals.

The order of the class tokens is unimportant, and you can specify each token more than once.

*funcList*            A comma separated list of function names.

If you don't specify this argument, the domain restrictions apply to all math library functions.

## Default

None

## Description

This indicates the domain on which a function is evaluated, and specifies that your program will function correctly if the functions specified in *funcList* do not produce standard conforming results on the number classes.

### 3.3.2 New `-vec-report7` Compiler Option (13.1.0)

The `-vec-report7` option provides additional details about vectorization of loops, including statistics and metrics about loads, stores, type conversions and more. This information can be useful in understanding how much improvement might be expected from vectorization of a given loop.

The information displayed from this option can also be processed by a new vectorization analysis tool available from <http://intel.ly/XeSkW6>

## 3.4 Other Changes and Notes

### 3.4.1 Establishing the Compiler Environment

The `compilervars.sh` script is used to establish the compiler environment.

The command takes the form:

```
source <install-dir>/bin/compilervars.sh argument
```

Where *argument* is either `ia32` or `intel64` as appropriate for the architecture you are building for. Establishing the compiler environment also establishes the environment for the Intel® Debugger, Intel® Performance Libraries and, if present, Intel® C++ Compiler.

## 3.5 Known Issues

### 3.5.1 Coarray Issues

For a list of known issues with Fortran 2008 Coarray support, see [Coarray Known Issues](#).

Intel® Fortran Composer XE 2013 for Linux\* Installation Guide and Release Notes

## 3.6 Coarrays

No special procedure is necessary to run a program that uses coarrays in a shared-memory configuration; 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 product (optional) 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 `-coarray-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`.

### 3.6.1 How to Debug a Coarray Application

The following instructions describe how to debug a Coarray application.

1. Add a stall loop to your application before the area of code you wish to debug, e.g.:

```
LOGICAL VOLATILE :: WAIT_FOR_DEBUGGER
LOGICAL, VOLATILE :: TICK
:
DO WHILE (WAIT_FOR_DEBUGGER)
  TICK = .NOT. TICK
END DO
! Code you want to debug is here
!
```

The use of `VOLATILE` is required to ensure that the loop will not be removed by the compiler. If the problem is only found on one image, you can wrap the loop in `IF (THIS_IMAGE() .EQ. 4) THEN` or the like.

2. Compile and link with debug enabled (`-g`).
3. Create at least  $N+1$  terminal windows on the machine where the application will be running, where  $N$  is the number of images your application will have.
4. In a terminal window, start the application.  
`linuxprompt> ./my_app`
5. In each of the other terminal windows, set your default directory to be the same as the location of the application executable. Use the `ps` command in one of the windows to find out which processes are running your application:

```
linuxprompt> ps -ef | grep 'whoami' | grep my_app
```

There will be several processes. The oldest is the one you started in step 4 – it has run the MPI launcher and is now waiting for the others to terminate. Do not debug it.

The others will look like this:

```

<your-user-name> 25653 25650 98 15:06 ? 00:00:49 my_app
<your-user-name> 25654 25651 97 15:06 ? 00:00:48 my_app
<your-user-name> 25655 25649 98 15:06 ? 00:00:49 my_app

```

The first number is the PID of the process (e.g., 25653 in the first line).

Call the PIDs of these N processes running "my\_app" P1, P2, P3 and so on.

6. In each window other than the first, start your debugger and set it to stop processes when attached:

```

linuxprompt> idb -idb
(idb) set $stoponattach = 1

```

or

```

linuxprompt> gdb

```

7. Attach to one of the processes (e.g. to P1 in window 1, to P2 in window 2, etc.)

```

(idb) attach <P1> my_app

```

or

```

(gdb) attach <P1>

```

8. Get execution out of the stall loop:

```

(idb) assign WAIT_FOR_DEBUGGER = .FALSE.

```

or

```

(gdb) set WAIT_FOR_DEBUGGER = .false.

```

9. You can now debug.

If you are using idb, you can use the multiprocess capability of idb to have only one debugger window instead of N. First, attach to each process and get out of the loop (steps 7 and 8).

```

(idb) attach <P1> my_app
(idb) assign WAIT_FOR_DEBUGGER = .FALSE.
(idb) attach <P2> my_app
(idb) assign WAIT_FOR_DEBUGGER = .FALSE.
(idb) attach <P3> my_app
(idb) assign WAIT_FOR_DEBUGGER = .FALSE.

```

Use the "process" command to switch debugging focus from one process to another:

```
(idb) process <Pn>
```

Processes not focused on will remain in the state they were left in: with breakpoints and watchpoints set but not running.

### 3.6.2 Compiler Option to Improve Coarray Performance (13.0.1)

Work is ongoing to improve performance of coarray applications. Some of this work is present in the 13.0.1 (Composer XE 2013 Update 1) compiler but is disabled by default. You can enable the optimizations implemented so far by adding the compile option:

```
-switch coarray_opts
```

when you compile your application. At present, this improves the performance of copying coarray values from another image. Not all applications may see a significant improvement when using this option. We encourage you to try this option and to [let us know](#) if it introduces errors into your application.

In a future major version, these optimizations will be enabled by default and the option shown above will be removed.

### 3.6.3 Coarray Known Issues

The following features are known not to work completely in this version:

- Accessing another image's value of an ALLOCATABLE or POINTER component of a derived-type coarray. Some forms of this work, some do not.

## 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
- ERRMSG 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
- The keyword MODULE may be omitted from MODULE PROCEDURE in a generic interface block when referring to an external procedure
- 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

The following is a partial list of Fortran 2003 features that are unimplemented or are known not to work in this release.

- User-defined derived type I/O
- Parameterized derived types
- Transformational intrinsics, such as `MERGE` and `SPREAD`, in initialization expressions

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 `ATOMIC_DEFINE`, `ATOMIC_REF`, `IMAGE_INDEX`, `LCBOUND`, `NUM_IMAGES`, `THIS_IMAGE`, `UCBOUND`
- `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.

## 4 Intel® Debugger (IDB)

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

### 4.1 Support Deprecated for Intel® Debugger

In a future major release of the product, the Intel® Debugger may be removed. This would remove the ability to use:

- The `idbc` command line debugger
- The `idb` GUI based debugger

### 4.2 Setting up the Java\* Runtime Environment

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

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

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

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

### 4.3 Starting the Debugger

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

```
idb
```

or

`idbc`

as desired.

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

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

## 4.4 Additional Documentation

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

Context-sensitive help is also available in most debugger dialogs, indicated by a “?” button.

## 4.5 Debugger Features

## 4.6 Known Issues and Changes

### 4.6.1 Coarray elements cannot be viewed.

The IDB Debugger cannot view coarray elements. Please refer to ['How to Debug a Coarray Application'](#) where a workaround is described.

### 4.6.2 Signals Dialog not working Signals Dialog not working

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

### 4.6.3 Resizing GUI

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

### 4.6.4 \$cdir, \$cwd Directories

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

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

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

### 4.6.5 info stack Usage

The gdb mode debugger command `info stack` does not currently support negative frame counts the way gdb does, for the following command:

```
info stack [num]
```

A positive value of `num` prints the innermost `num` frames, a zero value prints all frames, and a negative value prints the innermost `-num` frames in reverse order.

#### 4.6.6 `$stepg0` Default Value Changed

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

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

#### 4.6.7 SIGTRAP error on some Linux\* Systems

On some Linux distributions (e.g. Red Hat Enterprise Linux Server release 5.1 (Tikanga)) a SIGTRAP error may occur when the debugger stops at a breakpoint and you continue debugging. As a workaround you may define the SIGTRAP signal as follows on command line:

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

Caveat: With this workaround all SIGTRAP signals to the debuggee are blocked.

#### 4.6.8 `idb` GUI cannot be used to debug MPI processes

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

#### 4.6.9 Thread Syncpoint Creation in GUI

While for plain code and data breakpoints the field "Location" is mandatory, thread syncpoints require both "Location" and "Thread Filter" to be specified. The latter specifies the threads to synchronize. Please note that for the other breakpoint types this field restricts the breakpoints created to the threads listed.

#### 4.6.10 Stack Alignment for IA-32 Architecture

Due to changes in the default stack alignment for the IA-32 architecture, the usage of inferior calls (i.e. evaluation of expressions that cause execution of debuggee code) might fail. This can cause as well crashes of the debuggee and therefore a restart of the debug session. If you need to use this feature, make sure to compile your code with 4 byte stack alignment by proper usage of the `-falign-stack=<mode>` option.

#### 4.6.11 GNOME Environment Issues

With GNOME 2.28, debugger menu icons may not being displayed by default. To get the menu icons back, you need to go to the "System->Preferences->Appearance, Interface" tab and

enable, "Show icons in menus". If there is not "Interface" tab available, you can change this with the corresponding `GConf` keys in console as follows:

```
gconftool-2 --type boolean --set /desktop/gnome/interface/buttons_have_icons true
```

```
gconftool-2 --type boolean --set /desktop/gnome/interface/menus_have_icons true
```

#### 4.6.12 Accessing Online-Help

On systems where the Online-Help is not accessible from the IDB Debugger GUI Help menu, you can access the web-based debugger documentation from <http://intel.ly/ng91IO>

#### 4.6.13 Debugger crashes if \$HOME not set on calling shell

The debugger will end with a "Segmentation fault" if no \$HOME environment variable is set on the shell the debugger is started from.

#### 4.6.14 Command line parameter `--parallel` not supported

The debugger command line parameter `--parallel` is not supported on the shell command prompt nor on the Console Window of the Debugger GUI.

#### 4.6.15 Command line parameter `--idb` and `--dbx` not supported

The debugger command line parameters `--idb` and `--dbx` are not supported in conjunction with the debugger GUI.

#### 4.6.16 Core File Debugging

To be able to debug core files you need to start the debugger (command line debugger `idbc` or GUI debugger `idb`) with commandline options as follows:

```
idb|idbc <executable> <corefile>
```

<or>

```
idb|idbc <executable> -core <corefile>
```

Once started with a core file, the debugger is not able to debug a live process e.g. attaching or creating a new process. Also when debugging a live process, a core file cannot be debugged.

## 5 Intel® Xeon Phi™ Coprocessors

This section summarizes changes, new features and late-breaking news about developing for Intel® Xeon Phi™ coprocessors using Intel® Composer XE 2013 for Linux\*

### 5.1 Introduction

Intel® Fortran Composer XE 2013 supports development of applications that offload work to an Intel® MIC architecture coprocessor (Intel® Xeon Phi™ product family). These sections of code run on the Intel® Xeon Phi™ coprocessor if it is available. Otherwise, they run on the host CPU. Development of applications that run natively on Intel® Xeon Phi™ coprocessors is also supported.

Intel® Fortran Composer XE 2013 for Linux\* Installation Guide and Release Notes

This document uses the terms *coprocessor* and *target* to refer to the target of an offload operation.

## 5.2 Documentation

Documentation concerning the Intel® MIC architecture for Intel® Fortran Composer XE 2013 is currently undergoing change. For the latest documentation updates, please go to our web site at <http://intel.ly/MxPFYx>

## 5.3 Debugger

You can attach to code running on an Intel® Xeon Phi™ coprocessor or you can debug code offloaded from the CPU.

Use of the debugger on a remote system through SSH requires setting the DISPLAY environment variable to your local X display to minimize lag caused by SSH display forwarding.

The package contains command line versions of the Intel® Debuggers. They are called `idbc` (for the Intel® 64 architecture host) and `idbc_mic` (for the Intel® MIC architecture target).

Please note that the auto-attach feature is not supported in the command line versions of the debuggers.

## 5.4 Changes and Known Issues

This section corrects or adds to the product documentation.

### 5.4.1 \*MIC\* tag added to compile-time diagnostics

The compiler diagnostics infrastructure has been modified to add an additional offload \*MIC\* tag to the output message to allow differentiation from the Target (Intel® MIC Architecture) and the host CPU compilations. The additional tag appears only in the Target compilation diagnostics issued when offload directives are seen.

The new tag permits easier association with either the CPU or Target compilation.

### 5.4.2 Direct (native) mode requires transferring libiomp5.so to coprocessor

The Intel® Manycore Platform Software Stack (Intel® MPSS) does not include Intel® compiler libraries typically found under `/lib`.

When running applications in direct mode (i.e. on the coprocessor), users must first upload (via `scp`) a copy of any shared object libraries the application uses. For example, the OpenMP\* library (`<install_dir>/compiler/lib/mic/libiomp5.so`) should be copied to the coprocessor (device names will be of the format `micN`, where the first coprocessor will be named `mic0`, the second `mic1`, and so on) before running the application.

Failure to make this library available will result in a run-time failure, such as:

```
/libexec/ld-elf.so.1: Shared object "libiomp5.so" not found, required
by "sample"
```

Some applications may require uploading additional libraries.

### 5.4.3 Tuning Memory Allocation Performance

The following text replaces information on this topic in the product documentation.

For user-allocated data on the coprocessor, using large (2 MB) page allocations via `mmap`, instead of `malloc` or `ALLOCATE`, may improve application performance.

Not all applications benefit from using a larger page size. In general, the performance impact from a larger page size depends greatly on the data access pattern. If the application accesses multiple data structures that are allocated in different pages, having only limit TLB entries for 2 MB pages on the coprocessor can cause performance degradation.

The default page size is 4KB for `malloc` and `ALLOCATE`.

### 5.4.4 Stepping “A” Hardware Requires `-opt-streaming-stores never`

If your Intel® Xeon Phi™ coprocessor is hardware stepping “A”, you must use the `-opt-streaming-stores never` option when compiling your application as otherwise the compiler may generate instructions not supported by the hardware. Stepping “B” and later hardware support the new instructions.

### 5.4.5 Runtime errors or crashes when running an application built with the initial Intel® Composer XE 2013 product release with the offload libraries from a later update

There is a breaking binary compatibility change in the offload libraries for Intel® Composer XE 2013 Update 1 that will cause runtime errors or crashes if you use the Update 1 (or later) libraries with a binary built with the previous Intel Composer XE 2013 compiler. Examples of the errors you may observe in this situation are:

Error 1:

```
***Warning: offload to device #0 : failed
```

Error 2:

```
Segmentation fault (core dumped)
```

Error 3:

```
terminate called after throwing an instance of 'COIRERESULT'
terminate called recursively
```

Error 4:

```
CARD--ERROR:1 myoiPageFaultHandler: (nil) Out of Range!
CARD--ERROR:1 _myoiPageFaultHandler: (nil) switch to default signal
handle
CARD--ERROR:1 Segment Fault!
```

```
HOST--ERROR:myoiScifGetRecvId: Call recv() Header Failed ! errno = 104
^CHOST--ERROR:myoiScifSend: Call send() Failed! errno = 104
HOST--ERROR:myoiSend: Fail to send message!
HOST--ERROR:myoiBcastToOthers: Fail to send message to 1!
HOST--ERROR:myoiBcast: Fail to send message to others!
```

To resolve these issues, you should rebuild your application fully with the latest available update of Intel® Fortran Composer XE 2013.

#### 5.4.6 Non-Contiguous Array Sections May Not Be Passed to an Offload Region

As of compiler version 13.1 (Intel® Fortran Composer XE 2013 Update 2), the run-time library enforces a restriction that non-contiguous array sections may not be passed into an offload region. For example:

```
integer, pointer :: p(:)
integer, target :: t(20)

p => t(1:20:2)

...

!dir$ omp offload target(mic)
!$omp parallel do shared(p)
...

```

In this example, *p* is a non-contiguous array slice and is passed into the offload region. This will now result in the run-time error:

```
offload error: offload data transfer supports only a single contiguous
memory range per variable
```

Make sure that all variables passed into an offload region are contiguous.

#### 5.4.7 Environment Variable for Controlling Offload Behavior

Several additional environment variables are available for controlling offload behavior.

##### 5.4.7.1 MIC\_USE\_2MB\_BUFFERS

Sets the threshold for creating buffers with large pages. A buffer is created with the large pages hint if its size exceeds the threshold value.

Example:

```
// any variable allocated on a coprocessor that is equal to
// or greater than 100KB in size will be allocated in large pages.
setenv MIC_USE_2MB_BUFFERS 100k
```

#### 5.4.7.2 MIC\_STACKSIZE

Sets the size of the offload process stack for all Intel® Xeon Phi™ coprocessors used in the application. This is the overall stack size. Use `MIC_OMP_STACKSIZE` to modify the size of each OpenMP\* thread.

Example:

```
setenv MIC_STACKSIZE 100M      // Sets MIC stack to 100 MB
```

#### 5.4.7.3 MIC\_ENV\_PREFIX

This is the general mechanism to pass environment variable values to each Intel® Xeon Phi™ coprocessor.

The value of `MIC_ENV_PREFIX` sets the value of the prefix which is used to recognize environment variable values intended for coprocessors. For example,

```
setenv MIC_ENV_PREFIX MYCARDS
```

will use “MYCARDS” as the string that indicates that an environment variable is intended for a specific coprocessor.

Environment variable values of the form

`<mic-prefix>_<var>=<value>`

will send `<var>=<value>` to each coprocessor.

Environment variable values of the form `<mic-prefix>_<card-number>_<var>=<value>`

will send `<var>=<value>` to the coprocessor numbered `<card-number>`.

Environment variable values of the form

`<mic-prefix>_ENV=<variable1=value1|variable2=value2>`

will send `<variable1>=<value1>` and `<variable2>=<value2>` to each coprocessor.

Environment variable values of the form

`<mic-prefix>_<card-number>_ENV=<variable1=value1|variable2=value2>`

will send `<variable1>=<value1>` and `<variable2>=<value2>` to the coprocessor numbered `<card-number>`.

Examples:

```
setenv MIC_ENV_PREFIX PHI      // Defines the prefix to be used
setenv PHI_ABCD abcd           // Sets ABCD=abcd on all coprocessors
setenv PHI_2_EFGH efgh        // Sets EFGH=efgh on coprocessor 2
setenv PHI_VAR X=x|Y=y         // Sets X=x and Y=y on all coprocessors
setenv PHI_4_VAR P=p|Q=q       // Sets P=p and Q=q on coprocessor 4
```



### 5.4.8 OFFLOAD\_DEVICES

The environment variable `OFFLOAD_DEVICES` restricts the process to use only the coprocessors specified as the value of the variable. `<value>` is a comma separated list of physical device numbers in the range 0 to `(number_of_devices_in_the_system-1)`.

Devices available for offloading are numbered logically. That is `_Offload_number_of_devices()` returns the number of allowed devices and device indexes specified in the target specifier of an offload directive are in the range 0 to `(number_of_allowed_devices-1)`.

Example

```
setenv OFFLOAD_DEVICES "1,2"
```

### 5.4.9 Debugging and Intel® Debugger

#### 5.4.9.1 Using IDB with Intel® Many Integrated Core Architecture

When using the Intel® Debugger for Intel® Many Integrated Core architecture the following limitations apply:

- When debugging native coprocessor applications on the command line, the remote debug agent `idbserver_mic` is uploaded and started using scp/ssh. This implies that the user id used to start `idbc_mic` must also exist on the coprocessor. Unless passwordless authentication has been configured for this user id, scp and ssh will require a password being typed.
- When debugging heterogeneous applications on the command line, the offload process is started as root. Using `idbc_mic` with a different user id than root will cause the offload process to not be visible by the remote debug server `idbserver_mic`. The workaround is to launch the command line debugger `idbc_mic` as root. Alternatively the options `-mpm-launch=1 -mpm-cardid=<card-id>` can be added to the default launch options: `idbc_mic -mpm-launch=1 -mpm-cardid=<card-id> -tco -rconnect=tcpip:<cardip>:<port>`

#### 5.4.9.2 Safely ending offload debug sessions

To avoid issues such as orphan processes or stale debugger windows when ending offload applications, manually end the debugging session before the application is reaching its exit code. The following procedure is recommended for terminating a debug session.

- Manually stop a debug session before the application reaches the exit-code.
- When stopped, press the red stop button in the toolbar in the MIC-side debugger first. This will end the offloaded part of the application.
- Next, do the same in the CPU-side debugger.
- The link between the two debuggers will be kept alive. The MIC-side debugger will stay connected to the debug agent and the application will remain loaded in the CPU-side debugger, including all breakpoints that have been set.

- At this point, both debugger windows can safely be closed.

#### 5.4.9.3 MIC-side debugger asserts on setting source dirs

Setting source directories in the ABR debugger might lead to an assertion.

The assertion should not affect debugger operation. To avoid the assertion, don't use source directory settings. The debugger will prompt you to browse for files it cannot locate automatically.

## 6 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).

### 6.1 What's New in Intel® MKL 11.0 Update 2

- New Intel® MKL Extended Eigensolver:
  - Intel® MKL Extended Eigensolver is a high performance package for solving symmetric standard or a generalized symmetric-definite eigenvalue problems on matrices in dense, LAPACK banded, and sparse (CSR) formats. It is based on an innovative fast and stable numerical algorithm named Feast (see [Attributions](#) section below)
- BLAS:
  - Optimized [C/Z]HERK for native execution on the Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
  - Optimized BLAS Level-3 subroutine, ?SYMM (all precisions) for automatic offload (AO) on Intel® MIC Architecture
- Sparse BLAS:
  - Improved performance of 0-based DCSRMM significantly
- LAPACK:
  - Improved performance of parallel versions of ?(OR/UN)G(LQ/QL/QR/RQ) functions significantly
  - Optimized LU (?GETRF), Cholesky (?POTRF), and QR (?GEQRF) factorization functions for automatic offload on Intel® MIC Architecture
  - Improved LU and SMP Linpack performance for 60-cores on Intel® MIC Architecture
- ScaLAPACK:
  - Updated version to 2.0.2. New functions introduced include:
    - P?HSEQR: Nonsymmetric Eigenvalue Problem
    - P?SYEVR/P?HEEVR: MRRR (Multiple Relatively Robust Representations) algorithm
- FFT:
  - Improved performance of complex-to-complex power-of-2 1D and 2D FFTs on Intel® MIC Architecture

- Improved performance of real-to-complex power-of-two and odd size 1D FFTs on Intel® MIC Architecture
  - Added example demonstrating use of MKL FFT in Compiler Assisted Offload usage model for Intel® MIC Architecture with Intel® Fortran compiler
  - Decreased DFTI descriptor commit time on Intel® MIC Architecture
  - Added FFTW interface wrapper libraries support for Intel® MIC Architecture
- Cluster FFT:
  - Implemented transposed order in multidimensional Cluster FFT transforms, including FFTW2 wrappers
- VSL:
  - Supported ICDF (Inverse cumulative distribution function) method in VSL Lognormal RNG
  - Added “const” specifier to declarations of Data Fitting and VSL Summary Statistics functions
  - Improved performance of Wichmann-Hill BRNG on Intel® MIC Architecture
- Data Fitting:
  - Improved performance of df[d/s]Interpolate1D, df[d/s]InterpolateEx1D, df[d/s]SearchCells1D, df[d/s]SearchCellsEx1D routines for default/quasi-uniform partition, sorted interpolation sites in scalar (number of interpolation sites is 1) and vector cases for Intel® Xeon® processor X5570 and Intel® Xeon® processor E5-2600
  - Supported DF\_DISABLE\_CHECK\_FLAG parameter in dfiEditVal editor to improve performance for small number of interpolation sites (fewer than one dozen) by disabling checking of the correctness of parameters in Data Fitting routines
- Transposition:
  - Parallelized general out-of-place matrix transposition (mkl\_?omatcopy2), improving its performance significantly
- Service functions:
  - Added mkl\_peak\_mem\_usage function which provides information about peak memory amount used by Intel MKL Memory Allocator
  - Added mkl\_calloc and mkl\_realloc functions extending Intel® MKL Memory Allocator functionality to standard C library memory allocation API
- Enhanced SMP LINPACK with residual check:
  - It returns error code 1 if a failure is detected and prints conclusion if resulting residuals are ok to pass precision check or not. Please note that residuals might slightly vary from run-to-run on the same matrix if conditional numerical reproducibility mode is not turned on. The check ensures that results are reliable.

## 6.2 What's New in Intel® MKL 11.0 Update 1

- LAPACK

- Optimized ?(SY/HE)TRD, ?(OR/UN)M(LQ/QL/QR/RQ),?(OR/UN)GQR,?GE(QR/LQ/RQ/QL)F functions for native performance on Intel® MIC Architecture
  - Improved ?GETRF and SMP LINPACK benchmark native performance on Intel® MIC Architecture
  - Optimized ?GETRF, ?GEQRF, ?PORTF functions for automatic offload on Intel® MIC Architecture
- BLAS
  - Optimized [S/D/C/Z]SYMM for native execution on Intel® MIC Architecture
  - Improved DGEMM and double-precision Level 3 BLAS performance on AMD\* code name “Bulldozer” CPUs
- Sparse BLAS
  - Optimized CSRMV functionality for complex conjugate transpose and Hermitian cases
- PARDISO
  - Imaginary part of the diagonal values for Hermitian matrices are ignored
- Cluster FFT
  - Improved hybrid Cluster FFT (MPI + OpenMP\*) performance up to 2 times
  - A new Cluster FFT algorithm (Segment of Interest FFT) that uses less communication was implemented for 1D FFTs and it can be enabled by setting the environment variable "MKL\_CFFT\_SOI\_ENABLE" to "YES" or "1" — see more info in MKL documentation
- VSL
  - Added support of VSL\_SS\_METHOD\_FAST\_USER\_MEAN method for computation of descriptive Summary Statistics estimates given user-provided mean
  - Improved performance of VSL\_SS\_METHOD\_FAST method for computations of descriptive Summary Statistics estimates on Intel® Xeon® processor E5-2690 CPU
  - Improved performance of Summary Statistics algorithms for computation of raw and central moments, and variance-covariance estimates on Intel® MIC Architecture
  - Improved performance of MT2203 and WH BRNGs on Intel® MIC Architecture
- Transposition
  - Improved performance of Out-of-place transposition on 2nd generation Intel® Core™ microarchitecture (up to 7x)
- Service functions
  - Removed seven service functions with obsolete names (see more details at <http://intel.ly/OqbZEL>)

### 6.3 What's New in Intel® MKL 11.0

- Intel MKL now has support for Intel® Xeon Phi™ coprocessor based on the Intel® Many Integrated Core Architecture (Intel® MIC Architecture) on Linux\* only. There are three

Intel® Fortran Composer XE 2013 for Linux\* Installation Guide and Release Notes

Intel MKL usage models on Intel Xeon Phi coprocessor: automatic offload, compiler assisted offload and native execution. Most of Intel MKL has been ported to run natively on these coprocessors. A smaller number of functions have been optimized to automatically divide their computational work between the host and Intel Xeon Phi coprocessor, a feature called automatic offload (AO). Read the Intel MKL User's Guide for more information. Most standard Intel MKL functions run on Intel Xeon Phi coprocessor except the Poisson library, Iterative sparse solvers, and Trust region solvers.

- Conditional Bitwise Reproducibility (CBWR): New functionality in Intel MKL now allows you to balance performance with reproducible results by allowing greater flexibility in code branch choice and by ensuring algorithms are deterministic. See the Intel MKL User's Guide for more information. Refer to the CBWR KB Article (<http://intel.ly/P4yRXR>) for more information.
- Intel MKL also introduces optimizations using the new Intel® Advanced Vector Extensions 2 (AVX2) including the new FMA3 instructions. See the KB Article on support for Intel® AVX2 (<http://intel.ly/PVmq3h>) for more information.
- BLAS
  - Optimized [S/D/C/Z]GEMM, [S/D/C/Z]TRMM, [S/D/C/Z]TRSM, [S/D/C/Z]SYRK, [S/D]GEMV, [S/D]AXPY, [S/D]DOT for native execution and ?TRMM, ?TRSM, ?GEMM functions for automatic offload on the Intel® MIC Architecture
  - Improved DSYRK/SSYRK performance for 64-bit programs supporting Intel® Advanced Vector Extensions (Intel® AVX)
- Sparse BLAS
  - Optimized ?CSRMV, ?CSRMM, and ?CSRSMV (for unit diagonal case) for native execution on Intel® MIC Architecture
- LAPACK
  - Optimized [S/D]GETRF, [S/D]POTRF, [S/D]GEQRF, [S/D]GELQF, [S/D]GEQLF, and [S/D]GERQF for native execution on Intel® MIC Architecture
  - Introduced support for LAPACK version 3.4.1
- FFT
  - Optimized single- and double-precision real-to-complex and complex-to-complex one-, two-, and three-dimensional fast Fourier transforms for native execution on Intel® MIC Architecture
  - Added configuration parameter DFTI\_THREAD\_LIMIT which limits the number of threads per descriptor
  - Added support for 1D real-to-complex transforms with sizes given by 64-bit prime integers
- VML /VSL
  - Optimized complex SinCos and CIS functions for native execution on Intel® MIC Architecture
  - Optimized MT19937, MT2203, MRG32k3a BRNGs, and discrete Uniform and Geometric RNGs for native execution on Intel® MIC Architecture

- Improved performance of viRngGeometric on Intel® Advanced Vector Extensions (Intel AVX)
  - Implemented threading in Data Fitting Integrate1d function
- Transposition: Parallelized in-place transposition of square matrices with leading dimensions greater than the matrix size for single and double precisions improving its performance significantly
- Implemented local threading control function (mkl\_set\_num\_threads\_local) which increases flexibility in threading control
- The mklvars.\* script no longer sets \$FPATH in environment and no longer exports internal variable MKL\_TARGET\_ARCH (this change will not impact users as the Intel® compiler no longer requires these variables)
- Link Tool: Added Intel® MIC Architecture support
- Link Line Advisor:
  - Added Help-Me functionality for selecting architecture (IA-32/Intel® 64) and interface layer (LP64/ILP64)
  - Added Intel® MIC Architecture support

## 6.4 Deprecated and Removed Features

Please refer to the Intel® MKL Deprecations article (<http://intel.ly/LkZKGL>) for more information.

- Intel® MKL no longer supports execution on processors that do not support the Intel® SSE2 instruction set extensions (Intel® Pentium III and earlier.)
- Removed Intel MKL GNU Multiple Precision\* (GMP) function interfaces
- Disabled timing function mkl\_set\_cpu\_frequency() to perform useful work — use mkl\_get\_max\_cpu\_frequency(), mkl\_get\_clocks\_frequency(), and mkl\_get\_cpu\_frequency() as described in the Intel MKL Reference Manual
- Removed MKL\_PARDISO constant — used MKL\_DOMAIN\_PARDISO to specify the PARDISO domain with the mkl\_domain\_set\_num\_threads() function
- Removed special backward compatibility functions for convolution and correlation functions in Intel MKL 10.2 update 4
- Documentation:
  - The Intel MKL Reference Manual in HTML format is no longer available with the product
  - Man pages and Eclipse help integration are no longer provided

## 6.5 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://intel.ly/ptEfAP>

## 6.6 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](http://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. Petit, 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/>)

## 7 Disclaimer and Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL(R) PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY

APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

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

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

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

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or 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/](http://www.intel.com/products/processor_number/)

for details.

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

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

Copyright © 2013 Intel Corporation. All Rights Reserved.