1 Introduction

This document describes how to install the product, provides a summary of new and changed product features and 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 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
- Support added for Xcode® 4.5
- 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

- Support added for OS X* 10.8 “Mountain Lion”
- Support removed for OS X* 10.6 “Snow Leopard”
- Support added for Xcode® 4.3 and 4.4. Please note that the Command Line Tools component of Xcode 4.3 is required and is not installed by default
- Xcode-only installation is no longer available
- Installation and use on 32-bit versions of OS X* is no longer supported. All development must be done under a 64-bit version of OS X. Building of 32-bit applications is still supported.
- 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
• 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 OS X* includes the following components:
• Intel® Fortran Compiler XE 13.0.1 for building applications that run on Intel-based Mac* systems running the OS X* operating system
• Intel® Debugger 13.0.1
• Intel® Math Kernel Library 11.0 Update 1
• Integration into the Xcode* development environment (Limited Feature)
• On-disk documentation

1.3 System Requirements
• An Intel® 64 architecture based Apple* Mac* system
• 1GB RAM minimum, 2GB RAM recommended
• 2GB free disk space
• One of the following combinations:
  o OS X* 10.8 and Xcode* 4.5 or 4.4 or 4.3/SDK 10.8 with Command Line Tools component of Xcode installed
  o OS X* 10.7 and Xcode* 4.3/SDK 10.7 with Command Line Tools component of Xcode installed
• gcc* 4

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

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

1.5 Optimization Notice

<table>
<thead>
<tr>
<th>Optimization Notice</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>
1.6 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
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 will be using Xcode*, please make sure that a supported version of Xcode is installed. If you install a new version of Xcode in the future, you must reinstall the Intel Fortran Compiler afterwards.

You will need to have administrative or “sudo” privileges to install, change or uninstall the product.

If you received the compiler product on DVD insert the DVD. Locate the disk image file (xxx.dmg) on the DVD and double-click. If you received the compiler product as a download, double-click the downloaded file. When the disk image opens, double-click on the xxx.mpkg file to begin installation.

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 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

2.2 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.

2.3 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.

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 compiler 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
- debugger – symbolic link to the debugger directory
- man – symbolic link to the man directory
- Documentation – symbolic link to the Documentation directory
- Samples – symbolic link to the Samples directory

Each composer_xe_2013.<n>.<pkg> directory contains the following directories that reference a specific update of the Intel® Composer XE 2013 product:
• bin – all executables
• compiler – shared libraries and header files
• debugger – debugger files
• man – man pages
• Documentation – documentation files
• mkl – Intel® Math Kernel Library libraries and header files
• 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 compiler, no matter which version, the latest update of the Intel® Composer XE 2013 compiler, or a specific update. Most users will reference <install-dir>/bin for the compilevars.sh [.csh] script, which will always get the latest compiler installed. This method should remain stable for future releases.

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

1. Open Terminal and set default (cd) to any folder outside <install-dir>
2. Type the command:
   <install-dir>/composer_xe_2013.<n>./<pkg>./uninstall_fcompxe.sh
3. Follow the prompts

If you are not currently logged in as root you will be asked for the root password.

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 OS X* may be used in a build with version 13. Exceptions include:

• Sources that use the CLASS keyword to declare polymorphic variables and 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 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 and were compiled with versions earlier than 11.0 must be recompiled. The compiler will notify you if this issue is encountered.

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

In releases earlier than Intel® Fortran Composer XE 2011 (compiler version 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 12.0 and later compilers 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 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 New and Changed Directives

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

- ORDERED/END ORDERED
- SIMD VECTORLENGTHFOR

3.2.3 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.

3.2.4 Other 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.

3.3 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]
- -guide-profile=<file|dir>[,[file|dir],…]
- -openmp-link <library>
- -vec-report6

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

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 xxx is the package identifier and 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

### 3.6 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
- Default initialization for polymorphic 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

Fortran 2003 features not yet supported include:

• User-defined derived type I/O
• Parameterized derived types
• Transformational intrinsics, such as MERGE, 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)
• 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

Intel® Fortran Composer XE 2013 for OS X*
Installation Guide and Release Notes
• 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 intrinsic function NULL, 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.

Coarrays are not supported on OS X.

4 Intel® Debugger (IDB)

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 idb command line debugger.

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

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

```
ifort -g -o hello.exe hello.f90
```

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

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

```
ifort -c -g -o hello.o hello.f90
ifort -g -o hello.exe hello.o
```
Alternatively, users may use the compiler switch -save-temps to prevent the compiler from deleting the .o files it generates:

    ifort -g -save-temps -o hello.exe hello.f90

The debugger does not use the output of the “dsymutil” utility.

4.2 Known Problems

4.2.1 Compilation requirements for debugging 64-bit programs on OS X 10.7 “Lion”
OS X 10.7 “Lion” defaults to building 64-bit executables with Position Independent Executable (PIE) code. However, the Intel Debugger (IDB) does not currently support debugging executables built with PIE. To disable PIE, add the following options at the end of the ifort command line:

    -fpic -Wl,-no-pie

Note that the -g and -save-temps options are also required to build debuggable applications on all OS X versions.

4.2.2 Dwarf vs. Stabs Debug Formats
The debugger only supports debugging of executables whose debug information is in Dwarf2 format, and does not support the Stabs debug format. Use the -gdwarf-2 flag on the compile command to have gcc and g++ generate Dwarf output. The Intel® compilers (icc and ifort) produce Dwarf2 debug format with the -g flag.

4.2.3 Debug Info from Shared Libraries
The debugger does not read debug information from shared libraries. Therefore you cannot set a breakpoint to symbols like _exit which are part of a system library.

4.2.4 Non-local Binary and Source File Access
The debugger cannot access binary files from a network-mounted file system (such as NFS). The error message will look like this:

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

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

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

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

    ./hello.f90

    /auto/mount/site/foo/usr1/user_me/f_code/hello.f90

    (Cannot find source file hello.f90)
The file-path specified will be correct.

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

4.2.5 Local variables may not be visible
The linker on OS X (and subsequent versions) does not always issue definitions of local variables into the debug information in the executable. We do not have a characterization of when this occurs. The end result is that the variable is not visible or is visible but incorrectly evaluated.

The instances we have seen have involved local arrays in Fortran programs which were allocated in the .bss segment by the compiler. A work-around is to change the source to make the variable be global rather than local. In Fortran this is most easily done by putting the variable into a module or common block. Intel and Apple are working together to resolve this issue.

4.2.6 Printing Fortran REAL*16 variables
The debugger does not print the correct value for Fortran REAL*16 variables.

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

4.2.8 Debugging applications that exec
The $catchexecs control variable is not supported.

4.2.9 Fortran alternate entry points
Formal parameters of alternate entry points are not visible from within the debugger if they are not also formal parameters of the main entry point.

4.2.10 Snapshots
Snapshots are not yet supported as described in the manual.

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

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

Because the SIGBUS signal rather than the SIGSEGV signal is used by the debugger to implement watchpoints, you cannot create a signal detector which will catch a SIGBUS signal.
4.2.13 Fortran modules and commons
A globally defined Fortran module should be rescoped with a double percent (%%) when referred to. For example, to set a breakpoint in the subroutine bar contained in a globally defined module foo, do

(idb) stop in foo%%bar

Please refer to the following section in the manual for the rescoping syntax:

Looking Around the Code, the Data and Other Process Information >
Looking at the Data >
The print Command

If you try to access (print, etc.) a Fortran module or common using the name in the source code, the debugger may not be able to find it. As a workaround, you can try prepending '_' to the name. For example, in the source code, if you have a common called "com":

(idb) print _com

4.2.14 Graphical User Interface (GUI)
This version of the debugger does not support the GUI

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

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

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

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

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

4.2.19 Debugger variable "$threadlevel"
The manual’s discussion of the debugger variable "$threadlevel" says "On OS X*, the debugger supports POSIX threads, also known as pthreads." This sentence might be read as implying

* Intel® Fortran Composer XE 2013 for OS X*
Installation Guide and Release Notes
that other kinds of threads might be supported. This is not true; only POSIX threads are supported on OS X*.

4.2.20 Open File Descriptors Limitation
Because the debugger opens the .o files of a debuggee to read debug information, you should raise the open file limit.

OS* limits the number of open file descriptors to 256. You can increase this limit as follows:

\texttt{ulimit -n 2000}

Please use this command to increase the number of open descriptors before starting the debugger.

This is a workaround until the debugger can better share a limited number of open file descriptors over many files.

4.2.21 \$\texttt{cdir}, \$\texttt{cwd} Directories
\$\texttt{cdir} is the compilation directory (if recorded). This is supported in that the directory is set; but \$\texttt{cdir} is not itself supported as a symbol.

\$\texttt{cwd} is the current working directory. Neither the semantics nor the symbol are supported.

The difference between \$\texttt{cwd} and \'\.' is that \$\texttt{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.

4.2.22 \texttt{info stack} Usage
The gdb mode debugger command \texttt{info stack} does not currently support negative frame counts the way gdb does, for the following command:

\begin{verbatim}
   info stack [num]
\end{verbatim}

A positive value of \texttt{num} prints the innermost \texttt{num} frames, a zero value prints all frames, and a negative value prints the innermost $-\texttt{num}$ frames in reverse order.

4.2.23 \$\texttt{stepg0} Default Value Changed
The debugger variable \$\texttt{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:

\begin{verbatim}
(idb) set \$\texttt{stepg0} = 1
\end{verbatim}

4.2.24 Position Independent Code in 64-bit Programs under XCode 4.2 / OS X 10.7
The IDB debugger cannot debug 64-bit executables built with PIE (Position Independent Code). In order to debug code built under XCode 4.2/OS X 10.7, the compiler option \texttt{-Wl,-no-pie} (or linker option \texttt{-no-pie}) needs to be added in the build process.
5 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).

5.1 What’s New in Intel® MKL 11.0 Update 1

- 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
- 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)

5.2 What’s New in Intel® MKL 11.0

- 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/PVm3h) for more information.
- BLAS:
  - Improved DSYRK/SSYRK performance for 64-bit programs supporting Intel® Advanced Vector Extensions (Intel® AVX)
- LAPACK:
  - Introduced support for LAPACK version 3.4.1
• FFT:
  o Added configuration parameter DFTI_THREAD_LIMIT which limits the number of threads per descriptor
  o Added support for 1D real-to-complex transforms with sizes given by 64-bit prime integers
• VML /VSL:
  o Improved performance of viRngGeometric on Intel® Advanced Vector Extensions (Intel AVX)
  o 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 Line Advisor:
  o Added Help-Me functionality for selecting architecture (IA-32/Intel® 64) and interface layer (LP64/ILP64)

5.3 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.)
• PGI compilers are not supported for Intel® MKL running on Mac OS X* 10.8
• 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:
  o The Intel MKL Reference Manual in HTML format is no longer available with the product

5.4 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
5.5 Attributions
As referenced in the End User License Agreement, attribution requires, at a minimum, prominently displaying the full Intel product name (e.g. "Intel® Math Kernel Library") and providing a link/URL to the Intel® MKL homepage (www.intel.com/software/products/mkl) in both the product documentation and website.

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

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

The original versions of ScaLAPACK from which that part of Intel® MKL was derived can be obtained from http://www.netlib.org/scalapack/index.html. The authors of ScaLAPACK are 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.

6 Disclaimer and Legal Information
INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL(R) PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL’S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.
Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

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

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

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to:

http://www.intel.com/design/literature.htm

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


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

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

Copyright © 2012 Intel Corporation. All Rights Reserved.