Intel® Fortran Composer XE 2013 SP1 for OS X*
Installation Guide and Release Notes

Document number: 321416-005US
30 January 2014

Table of Contents

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 2 – February 2014

- Intel® Fortran Compiler updated to 14.0.2
  - Added –switch fe_debug_use_inherit internal command line switch
- Intel® Math Kernel Library update to 11.1 Update 2
- OS X* 10.9.1 supported
- KMP_DYNMIC_MODE Environment Variable Support for "asat" Deprecated

Update 1 – October 2013

- Intel® Fortran Compiler updated to 14.0.1
  - Added -assume std_value
  - Added -[a]xMIC-AVX512 compiler option
- Intel® Math Kernel Library updated to 11.1 Update 1
- Xcode* 5.0 supported

1.1.2 Changes Since Intel® Fortran Composer XE 2013

- Intel® Fortran Compiler updated to 14.0
- Intel® Math Kernel Library updated to 11.1
- GNU* Project Debugger (GDB)
- Intel® Debugger support deprecated
- An online version of the installer, where only required components are downloaded, is provided as an option
- Corrections to reported problems

1.2 Product Contents

*Intel® Fortran Composer XE 2013 SP1 Update 2 for OS X** includes the following components:

- Intel® Fortran Compiler XE 14.0.2 for building applications that run on Intel-based Mac* systems running the OS X* operating system
- GNU* Project Debugger (GDB*) 7.5
- Intel® Debugger 13.0
- Intel® Math Kernel Library 11.1 Update 2
- 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 of OS X*, Xcode* and the Xcode SDK:
  - OS X 10.9 and Xcode* 5.0
  - OS X 10.8 and Xcode* 4.6
- If doing command line development, the Command Line Tools component of Xcode* is required
- gcc* 4

Additional requirements to use GNU GDB*

- To use the provided GNU* GDB, Python* version 2.4, 2.6 or 2.7 is required.

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

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 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 Online Installer

The default electronic installation package now consists of a smaller installation package that dynamically downloads and then installs packages selected to be installed. This requires a working internet connection and potentially a proxy setting if you are behind an internet proxy. Full packages are provided alongside where you download this online install package if a working internet connection is not available.

2.2 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 Intel® Software Improvement Program.

2.3 Using a License Server

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

2.4 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
- ism – contains the Intel® Software Manager
- 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_sp1 directory
- composer_xe_2013_sp1 – directory containing symbolic links to subdirectories for the latest installed Intel® Composer XE 2013 SP1 product release
- composer_xe_2013_sp1.<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_sp1 directory contains the following directories that reference the latest installed Intel® Composer XE 2013 SP1 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_sp1.<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 SP1 compiler, or a specific update. Most users will reference <install-dir>/bin for the compilervars.sh [.csh]

script, which will always get the latest compiler installed. This method should remain stable for future releases.

2.5 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_sp1.<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 14. 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

Some language features may not yet be described in the compiler documentation. Please refer to the Fortran 2003 Standard (http://j3-fortran.org/doc/2003_Committee_Draft/04-007.pdf) if necessary.

3.2.1 Features from Fortran 2003

- User-Defined Derived Type I/O

3.2.2 Features from OpenMP* 4.0

The following directives, clauses and procedures, from OpenMP 4.0, are supported by the compiler. For more information, see the compiler documentation or the link to the OpenMP Technical Report.

SIMD Directives:

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

Other Directives:

- OMP PARALLEL PROC_BIND
- OMP TASKGROUP

3.2.2.1 KMP_DYNMIC_MODE Environment Variable Support for "asat" Deprecated

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

3.2.3 New and Changed Directives

The following compiler directives are new or changed in Intel® Composer XE 2013 SP1 – please see the documentation for details:

- [NO]FMA

3.3 New and Changed Compiler Options

Please refer to the compiler documentation for details.

- -assume std_value (14.0.1)

- -[a]xMIC-AVX512 (14.0.1)
- -fimf-domain-exclusion
- -fma
- -foptimize-sibling-calls
- -switch fe_debug_use_inherit (14.0.2)
- -vecabi
- -wrap-margin
- -xATOM_SSE4.2

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

3.3.1 New Option Affecting Fortran 2003 VALUE attribute

The Intel Fortran compiler's implementation of the Fortran 2003 VALUE attribute does not match the specification of the standard when used in a procedure that does not have the BIND(C) language binding specification. The compiler's default behavior is to treat the Fortran 2003 VALUE attribute the same as a DEC$ ATTRIBUTES VALUE directive causing the argument to be passed and received "by value". The standard specifies that a redefinable copy of the argument is to be passed instead. This incorrect behavior also prevents the use of the OPTIONAL attribute with VALUE. Note that if the procedure does have the BIND(C) language binding specification, then the implementation matches the standard and arguments with the VALUE attribute are passed and received by value.

In the version 14 compiler, the standard-conforming implementation is available but is not enabled by default, as this could cause problems for existing applications that assumed the previous implementation. To get the standard behavior add the /assume:std_value (Windows) or -assume std_value (Linux and OS X) compiler option. This option is not documented. When using Visual Studio on Windows, this option can be added under Command Line > Additional Options. If /standard-semantics (Windows) or -standard-semantics (Linux and OS X) is in effect, this implies std_value.

A future major release of the Intel Fortran compiler may change the default behavior for VALUE to match the standard.

3.3.2 New –[a]xMIC-AVX512 Compiler Option (14.0.1)

Optimizes for Intel(R) processors that support Intel(R) Advanced Vector Extensions 512 (Intel(R) AVX-512) instructions. May generate Intel(R) AVX-512 Foundation instructions, Intel(R) AVX-512 Conflict Detection instructions, Intel(R) AVX-512 Exponential and Reciprocal instructions, Intel(R) AVX-512 Prefetch instructions for Intel(R) processors, and the instructions enabled with CORE-AVX2.

3.3.3 New –switch fe_debug_use_inherit Internal Command Line Switch (14.0.2)

Examining the parent fields of an extended derived type in the gdb debugger currently requires that you also list the parent name. Add the internal command line switch –switch

fe_debug_use_inherit to your debug command line, and you will be able to use the abbreviated syntax to examine the parent field.

For example:

```
TYPE BASE
  integer Base_Counter
END TYPE BASE


TYPE, EXTENDS (BASE) :: Type2
END TYPE TYPE2


TYPE(Type2) :: Foo
```

It is legal Fortran to reference either Foo%Base_Counter or Foo%base%base_counter. Without the fe_debug_use_inherit switch, you cannot use the former form within gdb. Please note however, if you do set the fe_debug_use_inherit switch, you are unable to use the latter form within gdb.

This internal command line switch will not be supported in compiler version 15.0 as this feature will then be enabled by default.

3.4 Other Changes and Notes

3.4.1 Other Features

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

- ESTABLISHQQ library routine to specify that a user routine is to be called when the Fortran run-time library is about to report a run-time error. This routine is declared in module IFPORT.
- A command line option –[no-]wrap-margin, and an environment variable FORT_FMT_NO_WRAP_MARGIN, that control whether or not list-directed output begins a new record when the previous record would extend past column 80.
- A new predefined preprocessor macro __INTEL_COMPILER_UPDATE. This evaluates to an integer number for the update version of the compiler (0, 1, 2, etc.)
- New Environment variable FOR_FORCE_STACK_TRACE. When defined as 1, the compiler provides a traceback when any diagnostic message is issued at runtime. FOR_FORCE_STACK_TRACE overrides FOR_DISABLE_STACK_TRACE.

3.5 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, provided GNU* GDB (`gdb-ia`), Intel® Performance Libraries and, if present, Intel® C++ Compiler.

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
- User-Defined Derived Type I/O
- 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:

- 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
- 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 GNU* Project Debugger (GDB)

This section summarizes the changes, new features, customizations and known issues related to the GNU* GDB provided with Intel® Composer XE 2013 SP1.

4.1 Features

GNU* GDB provided with Intel® Composer XE 2013 SP1 is based on GDB 7.5 with enhancements provided by Intel. This debugger is planned to replace the Intel® Debugger in a future release. In addition to features found in GDB 7.5, there are several other new features:

- Support for Intel® Transactional Synchronization Extensions (Intel® TSX)
- Improved Fortran support

4.2 Pre-requisites

In order to use the provided GNU* GDB Python* version 2.4, 2.6 or 2.7 is required.

4.3 Using GNU* GDB

Instructions on how to use GNU* GDB can be found in the documentation

This debugger is designed to debug IA-32 or Intel® 64 applications natively. Its use is no different than with traditional GNU* GDB debuggers. There are some extensions, though, which can be found in the documentation.

4.4 Documentation

The documentation for the provided GNU* GDB can be found here:

```
<install-dir>/Documentation/en_US/debugger/gdb/gdb.pdf
<install-dir>/Documentation/en_US/debugger/ ↵
              gdb/eclmigdb_config_guide.pdf
```

5 Intel® Debugger (IDB)

Intel® Debugger (IDB) is available as host debugger for IA-32 and Intel® 64 applications.

5.1 Support Deprecated for Intel® Debugger

In a future major release of the product, the Intel® Debugger may be removed. This impacts all components and features described in this section.
New users should use the GNU* GDB debugger components instead.

5.2 Using Intel® Debugger

This debugger is designed to debug IA-32 or Intel® 64 applications natively. Instructions on how to use the Intel® Debugger is described in the documentation found here.

5.3 Documentation

Documentation for the Intel® Debugger can be found here:

```
<install-dir>/Documentation/en_US/debugger/
<install-dir>/Documentation/en_US/debugger/ ↵
              gdb/eclmigdb_config_guide.pdf
```

5.4 Compilation Requirements

5.4.1 Debug information stored in object files

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

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

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

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

1. Build the application in two steps, explicitly producing a `.o` file:

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

2. Use the compiler switch `-save-temps` to prevent the compiler from deleting the temporary `.o` files it generates:

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

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

5.4.2 Compilation requirements for debugging on OS X* 10.7 and higher (64-bit only)

Starting with OS X* 10.7 built 64-bit executables default to Position Independent Executable (PIE) code. However, the Intel® Debugger does not support debugging 64-bit executables built with PIE. To disable PIE, add `-Wl,-no_pie` to the compiler invocation. If the GNU* linker is used directly, use option `-no_pie`.

If in Xcode*, select “Don’t Create Position Independent Executables” under Build Settings. Note that the `-g` (and optionally `-save-temps` to save your object files) options are also required to build debuggable applications on all OS X* versions.

5.5 Known Issues

5.5.1 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, the you can try prepending '_' to the name. For example, in the source code, if you have a common called "com":

```
(idb) print _com
```

5.5.2 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 .

5.5.3 Printing Fortran REAL*16 variables

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

5.5.4 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.

5.5.5 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.

5.5.6 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.

5.5.7 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/c_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).

5.5.8 Debugging applications that fork

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

5.5.9 Debugging applications that exec

The $catchexecs control variable is not supported.

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

5.5.11 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.

5.5.12 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.

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

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

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

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

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

5.5.17 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.

5.5.18 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 that other kinds of threads might be supported. This is not true; only POSIX threads are supported on OS X*.

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

```
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.

5.5.20 `$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.

5.5.21 `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 one prints the innermost –num frames in reverse order.

5.5.22 `$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
```

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 11.1 Update 2

- Introduced support for Intel® Atom™ processors
- BLAS:
  - Improved performance of ?GEMM for m==1 or n==1 on all Intel architectures
  - Improved {S/D}GEMM single thread performance on small matrices for 64-bit processors supporting Intel® Advanced Vector Extensions (Intel® AVX) and Intel® Advanced Vector Extensions 2 (Intel® AVX2)
  - Improved DGEMV performance for 64-bit processors supporting Intel AVX2
  - Improved threaded performance of {S,D,C,Z}GEMV for notrans:n>>m and trans:m>>n on all Intel architectures
  - Improved DSYR2K performance for 64-bit processors supporting Intel AVX and Intel AVX2
  - Improved DTRMM performance on small matrices (A matrix size <= 10) for 64-bit processors supporting Intel AVX and Intel AVX2
  - Reduced stack usage for ZHEMM and ZSYRK
- LAPACK:
  - Improved performance of (S/D)SYRDB and (D/S)SYEV for large dimensions and UPLO=L when eigenvectors are needed
  - Improved performance of ?GELQF,?GELS and ?GELSS for underdetermined case (M
  - Improved performance of ?GEHRD,?GEEV and ?GEES
- Sparse BLAS:
  - Optimized SpMV kernels for Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instruction set
  - Improved Sparse BLAS level 2 and 3 performance for systems supporting Intel® Streaming SIMD Extensions 4.2 (Intel® SSE4.2), Intel AVX and Intel AVX2 instruction sets
- PARDISO:
  - Improved memory estimation of out-of-core portion size for reordering algorithm leading to improved factorization-solving step performance in OOC mode
- VML:
  - Added v[d|s]Frac function computing fractional part for each vector element
- VSL RNG:
  - Improved performance of MT2203 BRNG on CPUs supporting Intel AVX and Intel AVX2 instruction sets
- VSL Summary Statistics:
  - Added support for computation of group/pooled (VSL_SS_GROUP_MEAN/VSL_SS_POOLED_MEAN) mean estimates

6.2 What's New in Intel® MKL 11.1 Update 1

- Introduced support for Intel® AVX-512 instructions set with limited set of optimizations
- Added support for Visual Studio 2013 in the examples
- BLAS:
  - Improved performance of DSDOT, and added support for multiple threads, on all 64-bit Intel processors supporting Intel® Advanced Vector Extensions (Intel® AVX) and Intel® Advanced Vector Extensions 2 (Intel® AVX2)
  - Improved handling of denormals on the diagonal in *TRSM

    - Improved SGEMM performance for small N and large M and K on Intel® Many Integrated Core Architecture (Intel® MIC Architecture)
    - Improved parallel performance of *HEMM on all Intel processors supporting Intel® SSE4.2 and later
    - Improved parallel performance of 64-bit *SYRK/*HERK on all Intel processors supporting Intel® SSSE3 and later
    - Improved serial performance of 64-bit {D,S}SYRK on all Intel processors supporting Intel® SSE4.2 and later
    - Improved performance of DTRSM on Intel® MIC Architecture
    - Enhanced Intel® Optimized HPL Benchmark runmultiscript capabilities for Intel processors supporting Intel® AVX
    - Improved Intel® Optimized HPL Benchmark performance on Intel® MIC Architecture
- LAPACK
    - Decreased memory utilization for parallel LAPACK functions (OR/UN)M(QR/RQ/QL/LQ)
    - Decreased stack memory utilization in LAPACK functions
    - Improved performance of (S/D)SYRDB and (S/D)SYEV for large dimensions when eigenvalues are only needed
- ScaLAPACK
    - Updated PBLAS headers to mix default NETLIB and MKL complex datatypes
- DFT: Optimized complex-to-complex and real-to-complex transforms
- Transposition: Improved performance of mkl_?omatcopy routines on tall and skinny matrices
- DFTI interface and FFTW wrappers are now thread safe. Setting NUMBER_OF_USER_THREADS parameter when using MKL DFT from parallel regions became optional.

6.3 What's New in Intel® MKL 11.1

- Conditional Numerical Reproducibility : Introduced support for Conditional Numerical Reproducibility (CNR) mode on unaligned data
- Introduced MP LINPACK support for heterogeneous clusters - clusters whose nodes differ from each other, either by processor type or by having varying number of attached Intel® Xeon Phi™ coprocessors
- Introduced Clang compiler support on OS X*
- Improved performance of CNR=AUTO mode on recent AMD* systems
- BLAS:
    - Improved performance of [S/D]GEMV on all Intel processors supporting Intel® SSE4.2 and later
    - Optimized [D/Z]GEMM and double-precision Level 3 BLAS functions on Intel® Advanced Vector Extensions 2 (Intel® AVX2)
    - Optimized [Z/C]AXPY and [Z/C]DOT[U/C] on Intel® Advanced Vector Extensions (Intel® AVX) and Intel AVX2
    - Optimized sequential version of DTRMM on Intel MIC Architecture
    - Tuned DAXPY on Intel AVX2
- LAPACK:

    - Improved performance of (S/D)SYRDB and (S/D)SYEV for large dimensions when only eigenvalues are needed
    - Improved performance of xGESVD for small sizes like M,N<10
- VSL:
    - Added support and examples for mean absolute deviation
    - Improved performance of Weibull Random Number Generator (RNG) for alpha=1
    - Added support of raw and central statistical sums up to the 4th order, matrix of cross-products and median absolute deviation
    - Added a VSL example designed by S. Joe and F. Y. Kuo illustrating usage of Sobol QRNG with direction numbers which supports dimensions up to 21,201
    - Improved performance of SFMT19937 Basic Random Number Generator (BRNG) on Intel MIC Architecture
- DFT:
    - Improved performance of double precision complex-to-complex transforms on Intel MIC Architecture
    - Optimized complex-to-complex DFT on Intel AVX2
    - Optimized complex-to-complex 2D DFT on Intel® Xeon processor E5 v2 series (code named IvyTown)
    - Improved performance for workloads specific to GENE application on Intel Xeon E5-series (Intel AVX) and on Intel AVX2
    - Improved documentation data layout for DFTI compute functions
    - Introduced scaling in large real-to-complex FFTs
- Data Fitting:
    - Improved performance of df?Interpolate1D and df?SearchCells1D functions on Intel Xeon processors and Intel MIC Architecture
    - Improved performance of df?construct1d function for linear and Hermite/Bessel/Akima cubic types of splines on Intel MIC Architecture, Intel® Xeon® processor X5570 and Intel® Xeon® processor E5-2690
- Transposition
    - Improved performance of in-place transposition for square matrices
- Examples and tests for using Intel MKL are now packaged as an archive to shorten the installation time
- Link Tool and Link Line advisor: Added support for Intel MIC Architecture on Windows* OS

6.4 Notes

- Intel MKL now provides a choice of components to install. Components necessary for PGI compiler, Compaq Visual Fortran Compiler, SP2DP interface, BLAS95 and LAPACK95 interfaces, Cluster support (ScaLAPACK and Cluster DFT) and Intel MIC Architecture support are not installed unless explicitly selected during installation
- Unaligned CNR is not available for MKL Cluster components (ScaLAPACK and Cluster DFT)
- Examples for using Intel MKL with BOOST/uBLAS and Java have been removed from the product distribution and placed in the following articles:

- How to use Intel® MKL with Java*
- How to use BOOST* uBLAS with Intel® MKL

6.5 Known Issues

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

- An application built on OS X* and linked with libmkl_rt.so library where the first call to Intel® MKL was made in parallel section will crash with segfault or with either of these messages:

  “malloc: *** error for object xxxxx: pointer being freed was not allocated *** set a breakpoint in malloc_error_break to debug”

  or

  “malloc: *** error for object xxxxx: double free !!! *** set a breakpoint in malloc_error_break to debug”

  Workaround: call any Intel® MKL function before the parallel section

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) in both the product documentation and website.

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

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

The original versions of ScaLAPACK from which that part of Intel® MKL was derived can be obtained from http://www.netlib.org/scalapack/index.html. The authors of ScaLAPACK are L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley.

PARDISO in Intel® MKL is compliant with the 3.2 release of PARDISO that is freely distributed by the University of Basel. It can be obtained at http://www.pardiso-project.org.

Some FFT functions in this release of Intel® MKL have been generated by the SPIRAL software generation system (http://www.spiral.net/) under license from Carnegie Mellon University. The Authors of SPIRAL are Markus Puschel, Jose Moura, Jeremy Johnson, David Padua, Manuela Veloso, Bryan Singer, Jianxin Xiong, Franz Franchetti, Aca Gacic, Yevgen Voronenko, Kang Chen, Robert W. Johnson, and Nick Rizzolo.

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

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/

for details.

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

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