## Table of Contents

1 Introduction .................................................................................................................................................. 3  
1.1 Change History ....................................................................................................................................... 3  
1.2 Product Contents .................................................................................................................................. 4  
1.3 System Requirements .............................................................................................................................. 4  
1.4 Documentation ....................................................................................................................................... 6  
1.5 Samples .................................................................................................................................................. 6  
1.6 Japanese Language Support .................................................................................................................... 6  
1.7 Technical Support .................................................................................................................................. 7  
2 Installation .................................................................................................................................................... 7  
2.1 Pre-Installation Steps .............................................................................................................................. 7  
2.1.1 Configure Visual Studio for 64-bit Applications ................................................................................. 7  
2.1.2 Installation on Microsoft Windows Vista* or Microsoft Windows 7* .............................................. 8  
2.2 Installation ............................................................................................................................................... 8  
2.2.1 Installing the IMSL* Fortran Numerical Library* ............................................................................... 8  
2.2.2 Microsoft Visual Studio 2005 Premier Partner Edition Not Supported .............................................. 8  
2.2.3 Prompt for Administrator Permission with Microsoft Visual Studio 2005* ................................. 9  
2.3 Changing, Updating and Removing the Product ................................................................................... 9  
2.4 Installation Folders .................................................................................................................................. 9  
3 Intel® Visual Fortran Compiler .................................................................................................................... 10  
3.1 Compatibility .......................................................................................................................................... 11  
3.1.1 Incorrect Derived Type Layout for Type-Bound Procedures...............................................................11  
3.1.2 Removal of Inappropriate Hidden Arguments for BIND(C) Procedures ........................................... 11  
3.1.3 Change in Implementation of /Qopenmp-threadprivate:compat ..................................................... 12  
3.2 New and Changed Compiler Features .................................................................................................... 12
1 Introduction
This document describes how to install the product, provides a summary of new and changed product features and includes notes about features and problems not described in the product documentation.

1.1 Change History
This section highlights important changes in product updates. For a list of corrections to reported problems, please read Intel® Professional Edition Compilers 11.1 Fixes List.

Update 5
- Intel® Math Kernel Library updated to 10.2 Update 4
- Note added to Compatibility section with information about OpenMP* threadprivate variables.
- Corrections to reported problems

Update 4 (11.1.054)
- Intel® Math Kernel Library updated to 10.2 Update 3
- The Fortran project properties pages have been updated to allow enabling of the Intel® Parallel Debugger Extension
- Corrections to reported problems

Update 3 (11.1.051)
- Corrections to reported problems

Update 2 Revised (11.1.048)
- The cross-compilers (IA-32 to Intel® 64 and IA-32 to IA-64) were rebuilt to correct an issue where these compilers would not run on certain Windows 7* and Windows Server 2008* systems. Correctness of generated code is not an issue.
- Microsoft Windows 7* was added as a supported operating system

Update 2 (11.1.046)
- Hidden arguments are no longer used with BIND(C) routines. This may require source changes.
- Note added about new options /Qmkl, /Qimsl, and /QxAVX
- Note added about change in behavior for /warn:interface
• Note added about issue with diagnostics from command-line compiles on Intel® 64 architecture for filenames containing Japanese characters
• Intel® Parallel Debugger Extension is now supported
• Corrections to reported problems

Update 1 (11.1.038)

• Sources declaring or using derived types containing type-bound procedures must be recompiled.
• Note added about change in behavior of /Od.
• FORT_BLOCKSIZE and FORT_BUFFERCOUNT environment variables documented
• Corrections to reported problems

Product Release (11.1.035)

1.2 Product Contents
Intel® Visual Fortran Compiler Professional Edition 11.1 for Windows* includes the following components:

• Intel® Visual Fortran Compilers for building applications that run on IA-32, Intel® 64 and IA-64 architecture systems
• Intel® Assembler for IA-64 Architecture Applications
• Intel® Math Kernel Library 10.2 Update 4
• Integration into Microsoft® development environments
• Intel® Parallel Debugger Extension for Microsoft Visual Studio 2005 and 2008*
• Microsoft Visual Studio 2008 Shell and Libraries (not included with Student or Evaluation licenses nor in Compiler Suite products)
• Sample programs
• On-disk documentation

Intel® Visual Fortran Compiler Professional Edition with IMSL* for Windows* includes the above plus the IMSL® Fortran Numerical Library* from Visual Numerics*

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

• A PC based on an IA-32 or Intel® 64 architecture processor supporting the Intel® Streaming SIMD Extensions 2 (Intel® SSE2) instructions (Intel® Pentium® 4 processor or later, or compatible non-Intel processor), or based on an IA-64 architecture (Intel® Itanium®) processor
  o For the best experience, a multi-core or multi-processor system is recommended
• 1GB RAM (2GB recommended)
• 2GB free disk space required for all product features and all architectures
• To use the Microsoft Visual Studio development environment or command-line tools to build IA-32 or Intel® 64 architecture applications, one of:
  o Microsoft Visual Studio 2008* Standard Edition or higher with C++ and “X64 Compiler and Tools” components installed [1]
  o Microsoft Visual Studio 2005* Standard Edition or higher with C++ and “X64 Compiler and Tools” components installed [1]
  o Intel® Visual Fortran development environment based on Microsoft Visual Studio 2008 Shell (included with some license types of Intel® Fortran Compiler) [2]
• To use the Microsoft Visual Studio development environment or command-line tools to build IA-32 architecture applications, one of:
  o Microsoft Visual Studio .NET 2003* with C++ component installed [3]
  o Microsoft Visual C++ .NET 2003* [3]
• To use the Microsoft Visual Studio development environment or command-line tools to build IA-64 architecture applications, one of:
• To use command-line tools only to build IA-32 architecture applications, one of:
• To use command-line tools only to build Intel® 64 architecture applications, one of:
  o Microsoft Windows Software Development Kit Update for Windows Vista*
  o Microsoft Windows SDK for Windows 2008 and .NET Framework 3.5*
• To use command-line tools only to build IA-64 architecture applications:
  o Microsoft Windows SDK for Windows 2008 and .NET Framework 3.5*
• To read the on-disk documentation, Adobe Reader* 7.0 or later

Notes:

1. Microsoft Visual Studio 2005 and 2008 Standard Edition installs the “x64 Compiler and Tools” component by default – the Professional and higher editions require a “Custom” install to select this.
2. Intel® Visual Fortran development environment based on Microsoft Visual Studio 2008 Shell is included with Academic and Commercial licenses for Intel Visual Fortran
Compiler Professional Edition. It is not included with Evaluation or Student licenses, nor with “Compiler Suite” products that also include the Intel® C++ Compiler. This development environment provides everything necessary to edit, build and debug Fortran applications. Some features of the full Visual Studio product are not included, such as:

- Resource Editor (see ResEdit* (http://www.resedit.net/), a third-party tool, for a substitute)
- Automated conversion of Compaq* Visual Fortran projects
- Microsoft language tools such as Visual C++ or Visual Basic*


4. Microsoft Visual Studio is not supported for installation on IA-64 architecture systems

5. If you will be installing Microsoft Visual Studio 2008 Shell and you wish to also use Microsoft Visual C++ 2008 Express Edition (for separate access to the Microsoft C++ compiler), you must uninstall Visual C++ 2008 Express Edition before installing Visual Studio 2008 Shell along with Intel Visual Fortran Compiler. After the Fortran install is complete, you may reinstall Visual C++ 2008 Express Edition if desired. Please note that the Fortran and C++ compiler environments will be separate and not combined.

6. Development on an IA-64 architecture system supports building IA-64 architecture applications only.

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

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

1.4 Documentation

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

1.5 Samples

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

1.6 Japanese Language Support

Intel compilers provide support for Japanese language users. Error messages, visual development environment dialogs and some documentation are provided in Japanese in addition to English. By default, the language of error messages and dialogs matches that of
your operating system language selection. Japanese-language documentation can be found in the ja_JP subdirectory for documentation and samples.

If you wish to use Japanese-language support on an English-language operating system, or English-language support on a Japanese-language operating system, you will find instructions at http://software.intel.com/en-us/articles/changing-language-setting-to-see-english-on-a-japanese-os-environment-or-vice-versa-on-windows/

1.7 Technical Support
If you did not register your compiler during installation, please do so at the Intel® Software Development Products Registration Center. Registration entitles you to free technical support, product updates and upgrades for the duration of the support term.

For information about how to find Technical Support, Product Updates, User Forums, FAQs, tips and tricks, and other support information, please visit http://www.intel.com/software/products/support/

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

2 Installation

2.1 Pre-Installation Steps

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

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

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

To use Microsoft Visual Studio 2005/2008 Team System Edition to build applications to run on IA-64 architecture systems, follow the above steps and ensure that the box “Itanium Compiler and Tools” is checked.
2.1.2 Installation on Microsoft Windows Vista® or Microsoft Windows 7®
On Microsoft Windows Vista® or Microsoft Windows 7®, Microsoft Visual Studio.NET 2003* is not supported. Microsoft Visual Studio 2005* users should install Visual Studio 2005 Service Pack 1 (VS 2005 SP1) as well as the Visual Studio 2005 Service Pack 1 Update for Windows Vista, which is linked to from the VS 2005 SP1 page. After installing these updates, you must ensure that Visual Studio runs with Administrator permissions, otherwise you will be unable to use the Intel compiler. For more information, please see Microsoft's Visual Studio on Windows Vista page (http://msdn2.microsoft.com/en-us/vstudio/aa948853.aspx) and related documents.

Microsoft Visual Studio 2008 does not need an update for compatibility with Windows Vista or Windows 7, but you are encouraged to install the latest Microsoft Service Pack for Visual Studio 2008.

See also Prompt for Administrator Permission with Microsoft Visual Studio 2005*.

2.2 Installation
If you are installing the product for the first time, please be sure to have the product serial number available as you will be asked for it during installation. A valid license is required for installation and use.

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

If you received your product as a downloadable file, double-click on the executable file (.EXE) to begin installation. Note that there are several different downloadable files available, each providing different combinations of components. Please read the download web page carefully to determine which file is appropriate for you.

You do not need to uninstall previous versions or updates before installing a newer version – the new version will coexist with the older versions. If you want to remove older versions, you may do so before or after installing the newer one.

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

2.2.2 Microsoft Visual Studio 2005 Premier Partner Edition Not Supported
Microsoft Visual Studio 2005 Premier Partner Edition (VSPPE), as provided by Intel Visual Fortran 10.0, 10.1 and 11.0, is not supported by version 11.1. If you have VSPPE installed but not a supported Visual Studio .NET 2003 or Visual Studio 2008, the full product package will install Microsoft Visual Studio 2008 Shell.

If you do also have a 2003 or 2008 version of Visual Studio installed, installing version 11.1 will remove the previous compiler integration which will also remove integration from Visual Studio 2005 Premier Partner Edition. If you wish, you can reinstall the previous version’s integration.
into VSPPE by running its installer, selecting Modify, and selecting only the Visual Studio 2005 integration.

2.2.3 Prompt for Administrator Permission with Microsoft Visual Studio 2005*

If you are installing on Microsoft Windows Vista* or later versions of Microsoft Windows and are using Microsoft Visual Studio 2005, Windows may display a dialog similar to the following:

If this is displayed, it is important that you click the Continue button and leave the “Always show this message” box checked. If you select “Exit Visual Studio” instead, or do nothing (this message times out after two minutes), the compiler integration will not install completely.

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

2.3 Changing, Updating and Removing the Product

Use the Windows Control Panel “Add or Remove Products” applet to change which product components are installed or to remove the product.

When installing an updated version of the product, you do not need to remove the older version first. You can have multiple versions of the compiler installed and select among them. If you remove a newer version of the product you may have to reinstall the integrations into Microsoft Visual Studio from the older version.

2.4 Installation Folders

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

- C:\Program Files\Intel\Compiler\11.1\xxx
  - bin
    - ia32
    - ia32_intel64
    - ia32_ia64
Where xxx is the three-digit build number and the folders under `bin`, `include` and `lib` are used as follows:

- **ia32**: Files used to build applications that run on IA-32
- **intel64** and **em64t**: Files used to build applications that run on Intel® 64
- **ia64**: Files used to build applications that run on IA-64
- **ia32_intel64**: Compilers that run on IA-32 to build applications that run on Intel® 64
- **ia32_ia64**: Compilers that run on IA-32 (or Intel® 64) to build applications that run on IA-64

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

### 3 Intel® Visual Fortran Compiler

This section summarizes changes, new features and late-breaking news about the Intel® Visual Fortran Compiler.
3.1 Compatibility
In general, object code and modules compiled with earlier versions of Intel Visual Fortran (8.0 and later) may be used in a build with version 11.1. Exceptions include:

- Objects that contain or call procedures with the BIND(C) attribute may require source changes and recompilation.
- Objects built with the multi-file interprocedural optimization (/Qipo) option must be recompiled.
- Objects built for the Intel® 64 or IA-64 architectures 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 Incorrect Derived Type Layout for Type-Bound Procedures
The initial version 11.1 compiler incorrectly adds unused space to a derived type containing type-bound procedures. This error was corrected in version 11.1 Update 1. All sources declaring or using objects of such type, and which were compiled with the initial 11.1 compiler, must be recompiled with version 11.1 Update 1 or later.

3.1.2 Removal of Inappropriate Hidden Arguments for BIND(C) Procedures
Earlier versions of the Intel Fortran Compiler incorrectly passed or used hidden arguments for procedures with the BIND(C) attribute. For example, if a Fortran routine had a CHARACTER argument and had BIND(C) specified, the compiler would assume that two hidden arguments were passed for the function return address and length, plus another hidden argument for each character argument. While such hidden arguments are necessary and correct for non-interoperable (without the BIND(C) attribute) Fortran procedures, the Fortran standard prohibits them for interoperable procedures. The Fortran and C argument list must have a 1:1 correspondence.

In many cases where this problem occurs, the procedure does not meet the standard’s rules for interoperable procedures. For example, if an argument is of type CHARACTER, the length must be 1 – an array of such characters may be an argument. Arrays of any nature are not permitted as function return values for interoperable procedures. However, it is possible to create a standard-conforming interoperable procedure for which the compiler was passing or expecting hidden arguments.

In 11.1 Update 2, the compiler has been corrected to no longer pass or expect hidden arguments. If you wrote C code which assumed that such hidden arguments needed to be passed you will have to rewrite it and recompile. You may find that in some cases, functions will need to be converted to subroutines with the result variable passed as an actual argument. We apologize for the inconvenience, but it is important for correctness and portability to have this error repaired.
3.1.3 Change in Implementation of /Qopenmp-threadprivate:compat

All Fortran sources that were compiled with the /Qopenmp-threadprivate:compat option must be recompiled with 11.1 Update 5 or later. This option specifies that the code be interoperable with the Microsoft Visual C++ implementation of OpenMP threadprivate variables, but the earlier Fortran implementation was incorrect, leading to link-time errors or incorrect run-time behavior.

With the new implementation, it is required that the Fortran COMMON or module variable that is made threadprivate is initialized (DATA in a BLOCK DATA subprogram for COMMON or an initialization value for module variables), or that the program is linked with C code where the corresponding variable is declared as a struct with the __thread or __declspec(thread) attribute and is also declared as threadprivate.

3.2 New and Changed Compiler Features


3.2.1 Features from Fortran 2003

- Object-oriented features
  - CLASS declaration
  - SELECT TYPE construct
  - EXTENDS_TYPE_OF and SAME_TYPE_AS intrinsic functions
  - Polymorphic entities
  - Inheritance association
  - Deferred bindings and abstract types
  - Type inquiry intrinsic functions
- Type-bound procedures
  - TYPE CONTAINS declaration
  - ABSTRACT attribute
  - DEFERRED attribute
  - NON_OVERRIDABLE attribute
  - Note: GENERIC attribute and type-bound operators are not supported in this release
- Deferred-length character entities
- PUBLIC types with PRIVATE components and PRIVATE types with PUBLIC components
- 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
- 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 
  \texttt{/assume:}noold\_maxminloc is specified. Fortran 95 specified that the value was 
  processor-dependent and Intel Fortran returns 1 by default. Performance will be lower if 
  \texttt{/assume:}noold\_maxminloc is specified.

3.2.2 Other Changes

• When string length checking is in effect (\texttt{/check:}bounds), and a character object is 
  passed as an argument, the minimum of the passed length and the declared length in 
  the called procedure is used as an upper limit
• Input value items in the form of a LOGICAL constant, for example T or .F, are no longer 
  accepted during list-directed or namelist-directed input when the corresponding variable 
  in the I/O list is not LOGICAL. Similarly, when the I/O list variable is of type LOGICAL, 
  the corresponding input value must be in the form of a LOGICAL constant. The new 
  \texttt{/assume:}old\_logical\_ldio option can be used to restore the older behavior.
• Per-compilation control of floating point exception behavior (\texttt{/fpe-all})

3.3 New and Changed Compiler Options

Please refer to the compiler documentation for details

• \texttt{/assume:}\{no\}ieee\_fp\_flags
• \texttt{/assume:}\{no\}old\_logical\_ldio
• \texttt{/assume:}\{no\}old\_maxminloc
• \texttt{/fpe-all}
• \texttt{/hotpatch}
• \texttt{/Qdiag-enable:}sc\{-include\}
• \texttt{/Qdiag-enable:}sc\{-parallel\}
• \texttt{/Qimsl}
• \texttt{/Qmkll\{:lib\}}
• \texttt{/QxAVX}

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

3.3.1 \texttt{/Od} no longer implies \texttt{/Op}

In version 11.1, the \texttt{/Od} option for disabling optimizations no longer implies \texttt{/Op} for maximizing 
floating-point precision. The \texttt{/Op} switch is deprecated, so we recommend using an explicit \texttt{/fp} 
option for applications that are sensitive to floating-point precision changes.

3.3.2 \texttt{/warn:}interface now implies \texttt{/gen\_interface}

As of version 11.1, specifying \texttt{/warn:}interface enables both the generation and use of interfaces 
for the purpose of error checking. You no longer need to also specify \texttt{/gen\_interface} if 
\texttt{/warn:}interface is in effect.
3.4 Other Changes

3.4.1 Build Environment Command Script Change
The command window script used to establish the build environment changed in version 11.1 to allow the optional specification of the version of Microsoft Visual Studio to use. If you are not using the predefined Start menu shortcut to open a build environment window, use the following command to establish the proper environment:

"C:\Program Files\Intel\Compiler\11.1\xxx\Bin\ifortvars.bat" arch [vs]

Where xxx is the update number and arch is one of ia32, ia32_intel64, intel64, ia32_ia64, or ia64 as described above under Installation Folders. vs is optional and can be one of vs2008 or vs2005. If vs is not specified, the version of Visual Studio specified at installation time for command-line integration is used by default. Note that Microsoft Visual Studio .NET 2003 cannot be specified using the vs argument.

If you have installed the compiler into a different path, make the appropriate adjustments in the command.

Note: If the version of Visual Studio installed is Visual Studio 2008 Shell, do not specify the vs argument.

3.4.2 Instruction Set Default Changed to Require Intel® Streaming SIMD Extensions 2 (Intel® SSE2)
When compiling for the IA-32 architecture, /arch:SSE2 (formerly /QxW) is the default as of version 11.0. Programs built with /arch:SSE2 in effect require that they be run on a processor that supports the Intel® Streaming SIMD Extensions 2 (Intel® SSE2), such as the Intel® Pentium® 4 processor and non-Intel processors. No run-time check is made to ensure compatibility – if the program is run on a processor that does not support the instructions, an invalid instruction fault may occur. Note that this may change floating point results since the Intel® SSE instructions will be used instead of the x87 instructions and therefore computations will be done in the declared precision rather than sometimes a higher precision.

All Intel® 64 architecture processors support Intel® SSE2.

To specify the older default of generic IA-32, specify /arch:IA32

3.4.3 Optimization Reports Disabled by Default
As of version 11.1, the compiler no longer issues, by default, optimization report messages regarding vectorization, automatic parallelization and OpenMP threaded loops. If you wish to see these messages you must request them by specifying /Qdiag-enable:vec, /Qdiag-enable:par and/or /Qdiag-enable:openmp, or by using /Qvec-report, /Qpar-report and/or /Qopenmp-report.

Also, as of version 11.1, optimization report messages are sent to stderr and not stdout.
3.4.4 New Environment Variables to Control I/O

Version 11.1 supports two additional environment variables which can be used to affect I/O behavior when an application is run.

FORT_BLOCKSIZE lets you specify the default BLOCKSIZE value to be used when BLOCKSIZE= is omitted on the OPEN statement. Valid sizes are 0 to 2147467264. Sizes will be rounded up to the nearest 512-byte boundary. The default BLOCKSIZE value is now 128KB.

FORT_BUFFERCOUNT lets you specify the default BUFFERCOUNT value to be used when BUFFERCOUNT= is omitted on the OPEN statement. Valid values are 0 to 127. If 0 is specified, the default value of 1 will be used.

3.4.5 OpenMP* Libraries Default to “compat”

In version 10.1, a new set of OpenMP* libraries was added that allowed applications to use OpenMP code from both Intel and Microsoft compilers. These “compatibility” libraries can provide higher performance than the older “legacy” libraries. In version 11.x, the compatibility libraries are used by default for OpenMP applications, equivalent to /Qopenmp-lib:compat. If you wish to use the older libraries, specify /Qopenmp-lib:legacy.

The “legacy” libraries (libguide.lib, libguide40.lib, etc.) will be removed in a future release of the Intel compilers.

3.4.6 OpenMP* Libraries Default to Dynamic Linking

As of version 11.0, OpenMP applications link to the dynamic OpenMP libraries by default. To specify static linking of the OpenMP libraries, specify /Qopenmp-link:static.

3.4.7 Samples Provided as ZIP Archives

As of version 11.1, the compiler programming samples are provided as ZIP archives. Unpack each ZIP archive to a writable folder. All samples are now provided as Visual Studio* solutions; command-line build instructions are also provided. Please read the samples.htm file for more information.

3.4.8 Fortran Project File Compatibility

The Fortran project file (.vfproj) format changed in version 11.0. If you open a project created with an earlier version of Intel Visual Fortran, you will get a message indicating that the project needs to be converted. The old project is saved with a file type of .vfproj.old. A version 11.x project cannot be used by an earlier version of the Fortran integration (but you can use older versions of the compiler that you have installed through Tools > Options > Intel Fortran > Compilers.)

3.4.9 New Library File ifmodintr.lib

Version 11.1 introduces a new compiled code support library file ifmodintr.lib. This is automatically referenced when you use an intrinsic module that contains procedures or data structures, such as ISO_C_BINDING or IEEE_ARITHMETIC. If you have set the linker option to ignore default library directives, you may need to add this library to your build options. Note that this static library will be used even if you are building against the DLL support libraries.
3.4.10 OpenGL* AUX Library Not Linked When Using Module IFOPNGL

Intel® Visual Fortran supplies module IFOPNGL which provides declarations for OpenGL* routines, types and constants. In the past, Microsoft has supported the OpenGL AUX library through glaux.lib. As of Microsoft Visual Studio 2008, this library is no longer provided.

Module IFOPNGL still contains the declarations for the AUX routines but no longer causes glaux.lib to be linked in automatically. If your application uses the AUX routines, and you are using Visual Studio 2005 or earlier, you will need to add both glaux.lib and advapi32.lib to the list of libraries referenced by your application. This can be done by naming them on the project property Linker > Input > Additional Dependencies.

If you are using Visual Studio 2008, you can copy glaux.lib from a Visual Studio 2005 installation, or rewrite your program to not use the AUX routines. Intel Visual Fortran provides OpenGL samples AnimateGL and Rings which demonstrate coding that replaces some use of AUX routines.

3.5 Known Issues

3.5.1 /debug:parallel Option Not Yet Supported

The /debug:parallel option is documented but not yet supported. It will be supported in a subsequent product update and will be mentioned in the release notes for that update.

3.5.2 Error Viewing Documentation in Visual Studio .NET 2003

If you are using Microsoft Visual Studio .NET 2003 but have not installed the Microsoft MSDN Library feature, you will get an error “Help Is Not Installed for Visual Studio” when using the Help menu item for Intel® Visual Fortran Compiler Help, or when using F1 context-sensitive help. As a workaround, select Help > Contents and click OK on the error message box which is then displayed. You will then be able to access the product help in the Contents pane. This issue does not affect Microsoft Visual Studio 2005 or 2008.

3.5.3 Command-Line Diagnostic Issue for Filenames with Japanese Characters

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

3.5.4 Limited Support for Empty Derived Types

Fortran 2003 adds the ability to declare a derived type with no data components. The Intel compiler has limited support for these in the current release. These limitations will be lifted in a future release of the compiler. The limitations are as follows:

- When an object of derived type is declared, the type must have at least one data component. Extending an empty type is supported. For example:

```fortran
  type t
  end type
```
type, extends (t) :: t1
end type

type, extends (t1) :: t2
  integer i
end type

type, extends (t2) :: t3
end type

type (t) :: rec1 ! Not supported, type t is empty
type (t1) :: rec2 ! Not supported, type t1 is empty
type (t2) :: rec3 ! Supported, type t2 is not empty
type (t3) :: rec4 ! Supported, type t3 is not empty

An exception is that it is supported to declare a class object with an empty type, for example:

  class(t1) :: rec5

If an unsupported use of an empty type is seen, the compiler will issue the diagnostic:

Declaring an object with no data component fields is not yet supported

- Referencing a component that is an empty type is not supported. For example, assuming the declarations above, in:

  call sub(rec4%t3, rec4%t1, rec3%t)
  print *, rec3%t1, rec4%
  call sub2(rec3%t2, rec4%t2)

  the references to rec4%t3, rec4%t1, rec4%, rec3%t, and rec3%t are not supported. References to rec3%t2 and rec4%t2 will be supported. If an unsupported reference is seen, the compiler will issue the diagnostic:

  Accessing an empty type is not yet supported

- A type constructor for an empty type is not supported. Again assuming the declarations above, the type constructor t() is not supported. If an unsupported constructor is seen, the compiler will issue the diagnostic:

  A type constructor for an empty type is not yet supported
3.6  Fortran 2003 Feature Summary

The Intel Fortran Compiler supports many features that are new to the latest revision of the Fortran standard, 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
- 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")
- ASSOCIATE construct
- SELECT TYPE construct
- In all I/O statements, the following numeric values can be of any kind: UNIT=, IOSTAT=...
• NAMELIST I/O is permitted on an internal file
• Restrictions on entities in a NAMELIST group are relaxed
• Changes to how IEEE Infinity and NaN are represented in formatted input and output
• FLUSH statement
• WAIT statement
• ACCESS='STREAM' keyword for OPEN
• ASYNCHRONOUS keyword for OPEN and data transfer statements
• ID keyword for INQUIRE and data transfer statements
• POS keyword for data transfer statements
• PENDING keyword for INQUIRE
• The following OPEN numeric values can be of any kind: RECL=
• The following READ and WRITE numeric values can be of any kind: REC=, SIZE=
• The following INQUIRE numeric values can be of any kind: NEXTREC=, NUMBER=, RECL=, SIZE=
• Recursive I/O is allowed in the case where the new I/O being started is internal I/O that does not modify any internal file other than its own
• IEEE Infinities and NaNs are displayed by formatted output as specified by Fortran 2003
• BLANK, DECIMAL, DELIM, ENCODING, IOMSG, PAD, ROUND, SIGN, SIZE I/O keywords
• DC, DP, RD, RC, RN, RP, RU, RZ format edit descriptors
• In an I/O format, the comma after a P edit descriptor is optional when followed by a repeat specifier
• Rename of user-defined operators in USE
• INTRINSIC and NON_INTRINSIC keywords in USE
• IMPORT statement
• Allocatable dummy arguments
• Allocatable function results
• PROCEDURE declaration
• Procedure pointers
• ABSTRACT INTERFACE
• PASS and NOPASS attributes
• The COUNT_RATE argument to the SYSTEM_CLOCK intrinsic may be a REAL of any kind
• Execution of a STOP statement displays a warning if an IEEE floating point exception is signaling
• MAXLOC or MINLOC of a zero-sized array returns zero if the option -assume noold_maxminloc is specified.
• Type inquiry intrinsic functions
• COMMAND_ARGUMENT_COUNT intrinsic
• EXTENDS_TYPE_OF and SAME_TYPE_AS intrinsic functions
• GET_COMMAND intrinsic
• GET_COMMAND_ARGUMENT intrinsic
GET_ENVIRONMENT_VARIABLE intrinsic
IS_IOSTAT_END intrinsic
IS_IOSTAT_EOR intrinsic
MAX/MIN/MAXVAL/MINVAL/MAXLOC/MINLOC intrinsics allow CHARACTER arguments
MOVE_ALLOC intrinsic
NEW_LINE intrinsic
SELECTED_CHAR_KIND intrinsic
The following intrinsics take an optional KIND= argument: ACHAR, COUNT, IACHAR, ICHAR, INDEX, LBOUND, LEN, LEN_TRIM, MAXLOC, MINLOC, SCAN, SHAPE, SIZE, UBOUND, VERIFY
ISO_C_BINDING intrinsic module
IEEE_EXCEPTIONS, IEEE_ARITHMETIC and IEEE_FEATURES intrinsic modules
ISO_FORTRAN_ENV intrinsic module

Fortran 2003 features not yet supported include:

- Type-bound operators and the GENERIC binding for type-bound procedures
- User-defined derived type I/O
- Parameterized derived types

4 Intel® Math Kernel Library

This section summarizes changes, new features and late-breaking news about the Intel® Math Kernel Library (Intel® MKL) 10.2 Update 3 as part of Intel® Visual Fortran Compiler Professional Edition.

4.1 Changes in This Version

For further information on improvements in this and previous releases, see http://software.intel.com/en-us/articles/new-in-intel-mkl-10-2/

4.1.1 Intel® Math Kernel Library 10.2 Update 4

New Features

- Introduced the single precision complex absolute value function SCABS1
- Introduced the solver ?DTSVB for diagonally dominant tri-diagonal systems which is up to 2x faster than the general solver with partial pivoting (?GTSV)
- Added routines for factorization (?DTTRFB) and the forward/backward substitution (?DTTRSB) of the diagonally dominant tri-diagonal systems

Performance improvements

- FFTs
  - Enhanced performance for transforms which are a multiple of 8 or 13
  - Optimized 1D complex cluster FFTs for non-power-of-2 vector lengths
• VSL
  o Convolution and Correlation computations that require decimation show significant improvements (re-link required, see Known Issues)

4.1.2 Intel® Math Kernel Library 10.2 Update 3

Performance Improvements

• BLAS
  • Threaded the 32-bit OS versions of the following BLAS Level 1 and 2 functions for Intel® Core™ i7 processors and Intel® Xeon® processor 5300, 5400, and 5500 series: (D,S,C,Z)COPY, (D,S,C,Z)SWAP, (D,S,C,Z)AXPY, (S,C)ROT, (S,C)DOT, CDOTC, (D,S,C,Z)GEMV, (D,S,C,Z)TRMV, (S,C)SYMV, (S,C)SYR, (S,C)SYR2
  • Improved 32-bit and 64-bit OS versions of the following BLAS level 1 functions for Intel® Xeon® processors 5300, 5400, 5500: ZAXPY, ZSCAL, ZDOT(U,C), and (D,S)ROT
  • Improved DGEMM threading efficiency for matrices with many more rows than columns for Intel® Xeon® processor 5300

• LAPACK
  • Improved scalability of the following LAPACK functions: ?POTRF, ?GEBRD, ?SYTRD, ?HETRD, and ?STEDC divide and conquer eigensolvers

• FFTs
  • Updated underlying kernels to provide widespread performance improvements in FFTs
  • Improved threading of 3D FFTs when a small number of transforms are calculated with a single function call
  • Extended threading to small size multidimensional transforms

• VML
  • Further optimization for these VML functions on Intel® Xeon® processor 5500 series: v(s,d)Asin, v(s,d)Acos, v(s,d)Ln, v(s,d)Log10, vsLog1p, v[d:hypot

• VSL
  • Improved performance of viRngPoisson and viRngPoissonV random number generators

Usability and Interface Improvements

• Improved example programs for uBLAS, Java, FFTW3, LAPACK95, and BLAS95
• Some examples in the reference manual were removed where identical examples in source code form also appeared in the examples directory
• New 64-bit integer (ILP64) fftw_mpi interfaces for cluster FFTs

4.1.3 Intel® Math Kernel Library 10.2 Update 2
New Features

- LAPACK 3.2
  - 238 new LAPACK functions
  - Extra Precise Iterative Refinement
  - Non-Negative Diagonals from Householder QR factorization
  - High Performance QR and Householder Reflections on Low-Profile Matrices
  - New fast and accurate Jacobi SVD
  - Routines for Rectangular Full Packed format
  - Pivoted Cholesky
  - Mixed precision iterative refinement (Cholesky)
  - More robust DQDS algorithm
- Introduced implementation of the DZGEMM Extended BLAS function (as described at http://www.netlib.org/blas/blast-forum/). See the description of the \*gemm family of functions in the BLAS section of the reference manual.
- PARDISO now supports real and complex, single precision data

Usability/Interface improvements

- Sparse matrix format conversion routines which convert between the following formats:
  - CSR (3-array variation) ↔ CSC (3-array variation)
  - CSR (3-array variation) ↔ diagonal format
  - CSR (3-array variation) ↔ skyline
- Fortran95 BLAS and LAPACK compiled module files (.mod) are now included
  - Modules are pre-built with the Intel Fortran Compiler and are located in the include directory (see Intel® MKL User’s Guide for full path)
  - Source is still available for use with other compilers
  - Documentation for these interfaces can be found in the Intel® MKL User’s Guide
- The FFTW3 interface is now integrated directly into the main libraries
  - Source code is still available to create wrappers for use with compilers not compatible with the default Intel® Fortran compiler convention for name decoration
  - See Appendix G of the Reference Manual for information
- DFTI_DESCRIPTOR_HANDLE now represents a true type name and can now be referenced as a type in user programs
- Added parameter to Jacobi matrix calculation routine in the optimization solver domain to allow access to user data (see the description of the djacobix function in the reference manual for more information)
- Added an interface mapping calls to single precision BLAS functions in Intel® MKL (functions with “s” or “c” initial letter) to 64-bit floating point precision functions has been added on 64-bit architectures (See “sp2dp” in the Intel® MKL User Guide for more information)
• Compatibility libraries (also known as “dummy” libraries) have been removed from this version of the library

Performance improvements

• Further threading in BLAS level 1 and 2 functions for Intel® 64 architecture
  o Level 1 functions (vector-vector): (CS,ZD,S,D)ROT, (C,Z,S,D)COPY, and (C,Z,S,D)SWAP
    ▪ Increase in performance by up to 1.7-4.7 times over version 10.1 Update 1 on 4-core Intel® Core™ i7 processor depending on data location in cache
    ▪ Increase in performance by up to 14-130 times over version 10.1 Update 1 on 24-core Intel® Xeon® processor 7400 series system, depending on data location in cache
  o Level 2 functions (matrix-vector): (C,Z,S,D)TRMV, (S,D)SYMV, (S,D)SYR, and (S,D)SYR2
    ▪ Increase in performance by up to 1.9-2.9 times over version 10.1 Update 1 on 4-core Intel® Core™ i7 processor, depending on data location in cache
    ▪ Increase in performance by up to 16-40 times over version 10.1 Update 1 on 24-core Intel® Xeon® processor 7400 series system, depending on data location in cache
• Introduced recursive algorithm in 32-bit sequential version of DSYRK for up to 20% performance improvement on Intel® Core™ i7 processors and Intel® Xeon® processors in 5300, 5400, and 7400 series.
• Improved LU factorization (DGETRF) by 25% over Intel MKL 10.1 Update 1 for large sizes on the Intel® Xeon® 7460 Processor; small sizes are also dramatically improved
• BLAS *TBMV/*TBSV functions now use level 1 BLAS functions to improve performance by up to 3% on Intel® Core™ i7 processors and up to 10% on Intel® Core™2 processor 5300 and 5400 series.
• Improved threading algorithms to increase DGEMM performance
  o up to 7% improvement on 8 threads and up to 50% on 3,5,7 threads on the Intel® Core™ i7 processor
  o up to 50% improvement on 3 threads on Intel® Xeon® processor 7400 series.
• Threaded 1D complex-to-complex FFTs for non-prime sizes
• New algorithms for 3D complex-to-complex transforms deliver better performance for small sizes (up to 64x64x64) on 1 or 2 threads
• Implemented high-level parallelization of out-of-core (OOC) PARDISO when operating on symmetric positive definite matrices.
• Reduced memory use by PARDISO for both in-core and out-of-core on all matrix types
• PARDISO OOC now uses less than half the memory previously used in Intel MKL 10.1 for real symmetric, complex Hermitian, or complex symmetric matrices
• Parallelized Reordering and Symbolic factorization stage in PARDISO/DSS
• Up to 2 times better performance (30% improvement on average) on Intel® Core® i7 and Intel® Core™2 processors for the following VML functions: v(s,d)Round, v(s,d)Inv, v(s,d)Div, v(s,d)Sqrt, v(s,d)Exp, v(s,d)Ln, v(s,d)Atan, v(s,d)Atan2

• Optimized versions of the following functions available for Intel® Advanced Vector Extensions (Intel® AVX)
  o BLAS: DGEMM
  o FFTs
  o VML: exp, log, and pow
  o See important information in the Intel® MKL User’s Guide regarding the mkl_enable_instructions() function for access to these functions

4.2 Known Issues
A full list of the known limitations of this release can be found in the Knowledge Base for the Intel® MKL at http://software.intel.com/en-us/articles/known-limitations-in-intel-mkl-10-2

4.3 Notices
The following change is planned for future versions of Intel MKL. Please contact Technical Support if you have concerns:

• Content in the libraries containing solver in the filenames will be moved to the core library in a future version of Intel MKL. These solver libraries will then be removed.

4.4 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. Some FFT functions in this release of the Intel® MKL DFTI have been generated by the UHFFT software generation system under license from University of Houston. The Authors of SPIRAL are Markus Puschel, Jose Moura, Jeremy Johnson, David Padua, Manuela Veloso, Bryan Singer, Jianxin Xiong, Franz Franchetti, Aca Gacic, Yevgen Voronenko, Kang Chen, Robert W. Johnson, and Nick Rizzolo.

5 Intel® Parallel Debugger Extension

This section summarizes changes, new features and late-breaking news about the Intel Parallel Debugger Extension as part of Intel Visual Fortran Compiler.

5.1 New Feature

With 11.1 Update 2 of the Intel Visual Fortran Compiler, parallel debugging support of Fortran programs with the Intel Parallel Debugger Extension is added.

5.2 Known Limitations

- Fortran multi-dimension arrays are not displayed correctly and are not accepted in filter expressions.
- Fortran complex types are not displayed correctly.
- Filters cannot be set on Fortran arrays with custom array bounds.

- If you are using Microsoft Visual Studio 2005, there are three Intel-specific exceptions that must be enabled manually. Select Debug > Exceptions, expand the Win32 Exceptions tree, and enable items:
  a1a01db0 Intel Parallel Debugger Extension Exception 0
  a1a01db1 Intel Parallel Debugger Extension Exception 1
  a1a01db2 Intel Parallel Debugger Extension Exception 2

  This needs to be done once per project.

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

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

- If you are using Microsoft Visual Studio 2008 and debugging 64-bit applications, you must have Visual Studio 2008 Service Pack 1 installed.

  - You can debug 64-bit applications under Visual Studio 2005 and 2008 without Service Packs only if they are linked to the low memory area. If not linked to the low memory area, you will not see any events until the debuggee terminates. After termination, all events are displayed in the event window. In order to debug 64-bit applications properly, set the base address to 0x10000 in Project > Properties > Linker >
• Local variables are displayed as “???” in the Data Sharing Events window.

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

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

• Reentrant call detection stops in Disassembly view.

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

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

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

5.3 Documentation
Intel Parallel Debugger Extension Documentation can be accessed via the Help menu of Microsoft Visual Studio or by clicking the Help button of specific dialog boxes.

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 by visiting Intel's Web Site.

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 © 2010 Intel Corporation. All Rights Reserved.