

Intel® Visual Fortran Compiler Professional Edition 11.1 for Windows* Installation Guide and Release Notes

Document number: 321417-002US
27 April 2009

Table of Contents

1	Introduction	2
1.1	Product Contents	2
1.2	System Requirements.....	3
1.3	Documentation.....	5
1.4	Samples.....	5
1.5	Technical Support.....	5
2	Installation.....	5
2.1	Pre-Installation Steps.....	5
2.1.1	Configure Visual Studio for 64-bit Applications.....	5
2.1.2	Installation on Microsoft Windows Vista*	6
2.2	Installation	6
2.2.1	Installing the IMSL* Fortran Numerical Library*.....	6
2.2.2	Microsoft Visual Studio 2005 Premier Partner Edition Not Supported	6
2.3	Changing, Updating and Removing the Product	7
2.4	Installation Folders.....	7
3	Intel® Visual Fortran Compiler	8
3.1	Compatibility	8
3.2	New and Changed Compiler Features	9
3.2.1	Features from Fortran 2003	9
3.2.2	Other Changes	9
3.3	New and Changed Compiler Options.....	10
3.4	Other Changes and Known Issues.....	10
3.4.1	Build Environment Command Script Change	10

3.4.2	Instruction Set Default Changed to Require Intel® Streaming SIMD Extensions 2 (Intel® SSE2).....	10
3.4.3	Optimization Reports Disabled by Default.....	11
3.4.4	OpenMP* Libraries Default to “compat”.....	11
3.4.5	OpenMP* Libraries Default to Dynamic Linking.....	11
3.4.6	Fortran Project File Compatibility	11
3.4.7	New Library File <code>ifmodintr.lib</code>	11
3.4.8	OpenGL* AUX Library Not Linked When Using Module IFOPNGL.....	11
3.4.9	<code>debug:parallel</code> Option Not Yet Supported	12
3.4.10	Error Viewing Documentation in Visual Studio .NET 2003	12
3.4.11	Debugging Cannot Start After Install of Intel® Parallel Composer Beta	12
3.4.12	Samples Provided as ZIP Archives	13
3.4.13	Limited Support for Empty Derived Types.....	13
3.5	Fortran 2003 Feature Summary.....	14
4	Intel® Math Kernel Library	16
4.1	Change History	16
4.2	Known Limitations.....	18
5	Disclaimer and Legal Information.....	21

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 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
- Integration into Microsoft* development environments
- 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*

1.2 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
 - 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
- Microsoft Windows XP*, Microsoft Windows Vista*, Microsoft Windows Server 2003*, Microsoft Windows Server 2008* or Microsoft Windows HPC Server 2008* (embedded editions not supported)
 - Microsoft Windows Server 2008 or Windows HPC Server 2008 requires Microsoft Visual Studio 2008* SP1 or Visual Studio 2008 Shell with Visual Studio 2008 SP1 update applied. Other versions of Visual Studio listed below are not supported on Windows Server 2008 or Windows HPC Server 2008.
- To use the Microsoft Visual Studio development environment or command-line tools to build IA-32 or Intel® 64 architecture applications, one of:
 - Microsoft Visual Studio 2008* Standard Edition or higher with C++ and “X64 Compiler and Tools” components installed [1]
 - Microsoft Visual Studio 2005* Standard Edition or higher with C++ and “X64 Compiler and Tools” components installed [1]
 - 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:
 - Microsoft Visual Studio .NET 2003* with C++ component installed [3]
 - 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:
 - Microsoft Visual Studio 2008 Team System Edition with C++ and “Itanium Compiler and Tools” components installed [4] plus Microsoft Windows SDK for Windows 2008 and .NET Framework 3.5*
 - Microsoft Visual Studio 2005 Team System Edition with C++ and “Itanium Compiler and Tools” components installed [4]
- To use command-line tools only to build IA-32 architecture applications, one of:
 - Microsoft Visual C++ 2008* Express Edition [5]

- Microsoft Visual C++ 2005* Express Edition and Microsoft Windows SDK for Windows 2008 and .NET Framework 3.5*
- To use command-line tools only to build Intel® 64 architecture applications, one of:
 - Microsoft Windows Software Development Kit Update for Windows Vista*
 - Microsoft Windows SDK for Windows 2008 and .NET Framework 3.5*
- To use command-line tools only to build IA-64 architecture applications:
 - 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*
3. Microsoft Visual Studio .NET 2003 is not supported on Microsoft Windows Vista. Support for Microsoft Visual Studio .NET 2003 will be removed in a future release of the product.
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® Parallel Composer 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.

1.3 Documentation

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

1.4 Samples

Samples for each product component can be found in the `Samples` folder as shown under [Installation Folders](#).

1.5 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:

1. From Control Panel > Add or Remove Programs, select “Microsoft Visual Studio 2005” (or 2008) > Change/Remove. The Visual Studio Maintenance Mode window will appear. Click Next.
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*

On Microsoft Windows Vista*, Microsoft Visual Studio .NET 2003* is not supported. Microsoft Visual Studio 2005* users (including Microsoft Visual Studio 2005 Premier Partner Edition) 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.

When installing on Microsoft Windows Vista and with Microsoft Visual Studio 2005, you may see one or more warning boxes saying that there are compatibility issues with Visual Studio 2005. In some cases, these warning boxes may be hidden behind the installer window making it appear that the installation has stalled. Look in the Windows task bar for additional windows that require acknowledgement before proceeding. You may safely allow Visual Studio 2005 to run as part of the compiler install process – after installation is complete, be sure to install the two Service Pack 1 updates as described in the paragraph above.

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.

2.3 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
 - intel64
 - ia64
 - Documentation
 - include
 - ia32
 - intel64
 - ia64
 - lib
 - ia32
 - intel64
 - ia64
 - mkl
 - benchmarks
 - em64t
 - examples
 - ia32
 - ia64
 - include
 - interfaces
 - tests

- tools
 - Samples
 - setup_f

Where `xxx` is the three-digit update 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 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.

Note: In version 11.0, the IA-32 architecture default for code generation changed to assume that Intel® Streaming SIMD Extensions 2 (Intel® SSE2) instructions are supported by the processor on which the application is run. [See below](#) for more information.

Note: In version 11.0, the default for OpenMP applications is to link against the dynamic form of the “compatibility” OpenMP libraries. [See below](#) for more information.

3.2 New and Changed Compiler Features

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

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 `/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 `/assume:noold_maxminloc` is specified.

3.2.2 Other Changes

- When string length checking is in effect (`/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. The new `/assume:old_logical_ldio` option can be used to restore the older behavior.

- Per-compilation control of floating point exception behavior (/fpe-all)

3.3 New and Changed Compiler Options

Please refer to the compiler documentation for details

- /assume:[no]ieee_fp_flags
- /assume:[no]old_logical_ldio
- /assume:[no]old_maxminloc
- /fpe-all
- /hotpatch
- /Qdiag-enable:sc-include
- /Qdiag-enable:sc-parallel

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

3.4 Other Changes and Known Issues

3.4.1 Build Environment Command Script Change

The command window script used to establish the build environment changed in version 11.0. 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" arg1  
[arg2]
```

Where *xxx* is the update number and *arg1* is one of ia32, ia32_intel64, intel64, ia32_ia64, ia64 as described above under [Installation Folders](#). *arg2* is optional and is one of vs2005 or vs2008, indicating the desired version of Microsoft Visual Studio to support for command line integration. If *arg2* is omitted, the default is the version specified or latest version present when the compiler was installed. If you have installed the compiler into a different path, make the appropriate adjustments in the command.

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 certain AMD* 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 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.

3.4.5 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` .

3.4.6 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.)

3.4.7 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.

3.4.8 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.

3.4.9 /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.

3.4.10 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.

3.4.11 Debugging Cannot Start After Install of Intel® Parallel Composer Beta

If you previously had a beta version of Intel® Parallel Composer installed you may find that Microsoft Visual Studio will display an error message “Microsoft Visual Studio has encountered an internal error. ...” after starting or stopping debugging of an application. To solve this, please delete the following file:

```
C:\Documents and Settings\\Application  
Data\Microsoft\VisualStudio\9.0\windows.prf for Visual Studio 2008 on Windows  
XP
```

```
C:\Users\\AppData\Roaming\Microsoft\VisualStudio\9.0\windows  
.prf for Visual Studio 2008 on Windows Vista
```

```
C:\Documents and Settings\\Application  
Data\Microsoft\VisualStudio\8.0\windows.prf for Visual Studio 2005 on Windows  
XP
```

```
C:\Users\\AppData\Roaming\Microsoft\VisualStudio\8.0\windows  
.prf for Visual Studio 2005 on Windows Vista
```

You may need to enable viewing of hidden files and folders to see this file. To do this, open a folder window and select Tools > Folder Options. Click the View tab and select the option Files and Folders > Hidden files and folders > Show hidden files and folders. Click “Apply to All Folders” then click OK, OK.

Deleting this file will reset all Visual Studio windows to default configurations and resolve this problem.

3.4.12 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.

3.4.13 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:

```
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%t
call sub2(rec3%t2, rec4%t2)
```

the references to `rec4%t3`, `rec4%t1`, `rec4%t`, `rec3%t1`, 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
```

3.5 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.

4.1 Change History

- Performance Improvements in the BLAS:
 - 32-bit improvements

- 40-50% improvement for (Z,C)GEMM on Quad-Core Intel® Xeon® processor 5300 series
 - 10% improvement for all GEMM code on Quad-Core Intel® Xeon® processor 5400 series
 - 64-bit improvements
 - 2.5-3% improvement for DGEMM on 1 thread on Quad-Core Intel® Xeon® processor 5400 series
 - 50% improvement for SGEMM on the Intel® Core™ i7 processor family
 - 3% improvement for CGEMM on 1 thread on the Intel® Core™ i7 processor family
 - 2-3% improvement for ZGEMM on 1 thread on the Intel® Core™ i7 processor family
 - 30% improvement for right-side cases of DTRSM on the Intel® Core™ i7 processor family
- Improvements to the direct sparse solver (DSS/PARDISO):
 - The performance of out-of-core PARDISO was improved by 35% on average.
 - Support of separate backward/forward substitution for DSS/PARDISO has been added.
 - A new parameter for turning off iterative refinement for DSS interface has been introduced.
 - A new parameter for checking sparse matrix structure has been introduced for PARDISO interface.
- The capability to track the progress of a lengthy computation and/or interrupt the computation has been added via a callback function mechanism. A function called `mkl_progress` can be defined in a user application, which will be called regularly from a subset of the MKL LAPACK routines. See the LAPACK Auxiliary and Utility Routines chapter in the reference manual for more information. Refer to the specific function descriptions to see which LAPACK functions support the feature.
- Transposition functions have been added to Intel MKL. See the reference manual for further detail.
- The C++ `std::complex` type can now be used instead of MKL-specific complex types.
- An implementation of the Boost uBLAS matrix-matrix multiplication routine is now provided which will make use of the highly optimized version of DGEMM in the Intel MKL BLAS. See the User guide for more information.
- Improvements to the sparse BLAS:
 - Support for all data types (single precision, complex and double complex) has been added.
 - Routines for computing the sum and product of two sparse matrices stored, both stored in the compressed sparse row format have been added.
- The Vector Math Library functions, `CdfNorm`, `CdfNormInv`, and `ErfcInv`, have been optimized to achieve much improved performance.
- Performance improvement on the Intel® Core™ i7 processor family:
 - 3-17% improvement for the following VML functions: `Asin`, `Asinh`, `Acos`, `Acosh`, `Atan`, `Atan2`, `Atanh`, `Cbrt`, `CIS`, `Cos`, `Cosh`, `Conj`, `Div`, `ErfInv`, `Exp`, `Hypot`, `Inv`, `InvCbrt`, `InvSqrt`, `Ln`, `Log10`, `MulByConj`, `Sin`, `SinCos`, `Sinh`, `Sqrt`, `Tanh`.
 - 7-67% improvement for uniform random number generation.
 - 3-10% improvement for VSL distribution generators based on Wichmann-Hill, Sobol, and Niederreiter BRNGs (64-bit only).
- The configuration file functionality has been removed. See the user guide for alternative means to configure the behavior of Intel MKL.
- All hurdles to creation of DLLs from the static libraries have been removed.

- When functions in Intel MKL are called from an MPI program they will be run on 1 thread by default (i.e., in the absence of explicit controls).
- The following VML functions have been added: CdfNorm, CdfNormInv, and ErfcInv.
- The DftiCopyDescriptor function has been added.
- The LP64 interface of DSS/PARDISO now uses 64-bit addressing for internal arrays on 64-bit operating systems. This allows the direct solver to solve larger systems.
- The default OpenMP runtime library for Intel MKL has been changed from libguide to libiomp. See the User Guide in the doc directory for more information.
- Documentation updates:
 - The parallel BLAS (PBLAS) which support ScaLAPACK are now documented in the Intel MKL reference manual.
 - Added instructions for using example programs in Microsoft* Visual Studio to the User Guide.
 - MKL Documentation is now accessible from the Microsoft* Visual Studio 'Help' menu with F1 Help and Dynamic Help features provided in the code editor. For more information, see Intel MKL User's Guide.
- It is no longer possible to set environment variables during the installation process. Three script files, mklvars32.bat, mklvarsem64t.bat, and mklvars64.bat are available in the tools\environment directory to set the PATH, LIB, and INCLUDE environment variables at the command prompt.
- The optimized code paths for the Intel® Pentium® III processor have been removed from Intel MKL along with the associated processor specific dynamic link libraries. We continue to support the use of Intel MKL on this processor, but the default code path will be used and as a result performance may be reduced.
- The interval linear solver functions have been removed from MKL.
- Documentation updates:
 - The FFTW Wrappers for MKL Notes have been removed from the product package after their content was integrated into the Intel MKL Reference Manual (Appendix G).
 - New functions have been documented in the reference manual, and support for Boost uBLAS matrix-matrix multiplication has been described in the User Guide.

4.2 Known Limitations

Limitations to the sparse solver and optimization solvers:

- Sparse and optimization solver libraries functions are only provided in static form

Limitations to the FFT functions:

- Mode DFTI_TRANSPOSE is implemented only for the default case
- Mode DFTI_REAL_STORAGE can have the default value only and cannot be set by the DftiSetValue function (i.e. DFTI_REAL_STORAGE = DFTI_REAL_REAL)
- The ILP64 version of Intel® MKL does not currently support FFTs with any one dimension larger than $2^{31}-1$. Any 1D FFT larger than $2^{31}-1$ or any multi-dimensional FFT with any dimension greater than $2^{31}-1$ will return the "DFTI_1D_LENGTH_EXCEEDS_INT32" error message. Note that this does not exclude the possibility of performing multi-dimensional FFTs with more than $2^{31}-1$ elements; as long as any one dimension length does not exceed $2^{31}-1$

- Some limitations exist on arrays sizes for Cluster FFT functions. See mklman.pdf for a detailed description
- When a dynamically linked application uses Cluster FFT functionality, it is required to put the static Intel® MKL interface libraries on the link line as well. For example: `-WI,--start-group $MKL_LIB_PATH/libmkl_intel_lp64.a $MKL_LIB_PATH/libmkl_cdft_core.a -WI,--end-group $MKL_LIB_PATH/libmkl_blacs_intelmpi20_lp64.a -L$MKL_LIB_PATH -lmkl_intel_thread -lmkl_core -liomp5 -lpthread`

Limitations to the LAPACK functions:

- The ILAENV function, which is called from the LAPACK routines to choose problem-dependent parameters for the local environment, cannot be replaced by a user's version
- `second()` and `dsecnd()` functions may not provide correct answers in the case where the CPU frequency is not constant.
- As of version 10.0 the following two issues apply when linking to the dynamic libraries:
 - A user provided XERBLA will not be invoked if LAPACK is called with illegal input parameters. The default XERBLA will be used instead.
 - A user may encounter a segment violation if they call the LP64 interface to LAPACK with illegal parameters. This is because a request may be made to allocate a negative amount of memory.

Limitations to the Vector Math Library (VML) and Vector Statistical Library (VSL) functions:

- Usage of `mkl_vml.fi` may produce warning about TYPE ERROR_STRUCTURE length
- In case user needs to build custom DLL that contains references to Intel® MKL functions, the Intel® MKL DLL Builder Tool should be used. Other DLL build techniques are not supported

Limitations to the ScaLAPACK functions:

- The user can not substitute PJLAENV for their own version. This function is called by ScaLAPACK routines to choose problem-dependent parameters for the local environment.
- There are possible problems with getting global environment variables such as `MKL_BLACS_MPI` by `-genvlist` by `MPICH2`. In this case, try to set all necessary environment variables by using the control panel. From the System control panel select the "Advanced" tab and click the "Environment Variables" button.

Limitations to the ILP64 version of Intel® MKL:

- The ILP64 version of Intel® MKL does not contain the complete functionality of the library. For a full listing of what is in the ILP64 version refer to the user's guide in the doc directory.

Limitations to the Java examples:

- The Java examples don't work if the path to the JDK contains spaces. Please use quotes to set JAVA_HOME in those cases. For example: set JAVA_HOME="C:\Program Files\Java\jdk1.6.0_06"

The DHPL_CALL_CBLAS option is not allowed when building the hybrid version of MP LINPACK.

We recommend that /Od be used for the Intel® compilers when compiling test source code available with Intel® MKL. Current build scripts do not specify this option and default behavior for these compilers has changed to provide vectorization.

Limitations to dummy libraries:

- Dummy libraries cannot be used in #pragma constructions. Dummy libraries cannot be linked by Intel compiler as a driver. Please see Chapter 3 of User's Guide for more information

All VSL functions return an error status, i.e., default VSL API is a function style now rather than a subroutine style used in earlier Intel® MKL versions. This means that Fortran users should call VSL routines as functions. For example:

```
errstatus = vslnrggaussian(method, stream, n, r, a, sigma)
```

rather than subroutines:

```
call vslnrggaussian(method, stream, n, r, a, sigma)
```

Nevertheless, Intel® MKL provides a subroutine-style interface for backward compatibility. To use subroutine-style interface, manually include mkl_vsl_subroutine.fi file instead of mkl_vsl.fi by changing the line include 'mkl_vsl.fi' mkl.fi (in the include directory) with the line include 'mkl_vsl_subroutine.fi'. VSL API changes don't affect C/C++ users.

Memory Allocation: In order to achieve better performance, memory allocated by Intel® MKL is not released. This behavior is by design and is a one-time occurrence for Intel® MKL routines that require workspace memory buffers. Even so, the user should be aware that some tools may report this as a memory leak. Should the user wish, memory can be released by the user program through use of a function (MKL_FreeBuffers()) made available in Intel® MKL or memory can be released after each call by setting the environment variable MKL_DISABLE_FAST_MM (see User's Guide in the doc directory for more details). Using one of these methods to release memory will not necessarily stop programs from reporting memory leaks, and in fact may increase the number of such reports should you make multiple calls to the library thereby requiring new allocations with each call. Memory not released by one of the methods described will be released by the system when the program ends. To avoid this restriction disable memory management as described above.

Other: The GMP component is located in the solver library. For Intel® 64 and IA-64 platforms these components support only LP64 interface.

Using /MT when linking with multi-threaded Intel® MKL is recommended. Use of /MD with Microsoft* Visual C++* .NET 2003 may cause linking errors.

5 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 © 2009 Intel Corporation. All Rights Reserved.