White Paper

Steven Lionel
Systems and Solutions Group
Intel Corporation

Porting Applications from Compaq* Visual Fortran to Intel® Visual Fortran

Table of Contents

Executive Summary

Intel Visual Fortran Compiler for Windows* is the successor to Compaq Visual Fortran*, combining the technologies of the Intel and Compaq compilers. This paper introduces the necessary background for developers who are migrating to the Intel Visual Fortran Compilers from Compaq Visual Fortran. It discusses changes in behavior, and it highlights source and build changes that may be needed as part of that migration.

Introduction

Intel® Visual Fortran Compiler 10.0 for Windows* integrates with Microsoft developer tools to provide very high levels of optimization for the full spectrum of Intel® processor technologies. In addition to performance and industry-compatibility gains, this version provides full support for IA-32, Intel® 64 (formerly Intel® EM64T) and IA-64 (Intel® Itanium) architecture processors including the latest Intel multi-core processors, as well as support for previous generations of processors.

In most cases, you can rebuild existing Compaq Visual Fortran (CVF) applications with the new compilers without source changes, but some applications may need minor coding changes, and build methods may need minor adjustments. This paper describes the key differences you are likely to encounter. For additional details, refer to the appropriate compiler release notes and the compiler documentation.

Product Compatibility

If you already have installed Microsoft Visual Studio 2005* or Microsoft Visual Studio .NET 2003*, Intel Visual Fortran will integrate into that Visual Studio development environment allowing you to create, build and debug Fortran and mixed-language applications. If you do not have a supported version of Microsoft Visual Studio installed, Intel Visual Fortran will install Microsoft Visual Studio Premier Partner Edition* to provide a rich development environment for Fortran programming on IA-32 and Intel® 64 systems. (Note that some license types of Intel Visual Fortran do not provide Microsoft Visual Studio Premier Partner Edition. Please read the compiler's System Requirements for details.)

The Intel Visual Fortran product can coexist on a system with CVF, and you can continue using the older product if you wish. However, Intel Visual Fortran does not integrate with Microsoft Visual Studio 6 environment used by CVF, whereas CVF does not integrate with the 2002 and later versions of Microsoft Visual Studio. The Intel and Compaq products install into separate folder trees and use separate registry variables.

All Fortran sources must be recompiled with Intel Visual Fortran Compiler; you cannot use CVF-compiled objects, modules, or static libraries with Intel Visual Fortran Compiler. You can, however, use CVF-built dynamic link libraries (DLLs) with applications compiled with the Intel Visual Fortran Compiler, as long as you do not try to share input/output units across the two environments. Note also that third-party libraries built for use with CVF may not work with Intel Visual Fortran Compiler. Contact the library supplier for more information.

Common Features

Intel Visual Fortran Compiler supports all of the CVF language syntax, including extensions from Digital Equipment Corporation (DEC) Fortran and Microsoft Fortran PowerStation* 4. All CVF library routines are supported, including those from the QuickWin and Portability libraries, as are all of the system-interface modules. In most cases, a simple rebuild of the application with the Intel compiler is all that is needed. Source changes that you may need to make are described below.

Features Not Supported

Intel Visual Fortran supports all of the language syntax supported by CVF. However, some of the CVF product features are not supported by Intel Visual Fortran. These include the following:

- Compaq Extended Math Library. The Intel® Math Kernel Library or third-party libraries such as IMSL* and NAG* may be suitable alternatives. (Intel Math Kernel Library is provided with Intel Visual Fortran Compiler, Professional Edition and the IMSL library is provided with Intel Visual Fortran Compiler, Professional Edition with IMSL. Other products must be obtained separately.)

- Source Browser cross-reference tool.
- Format editor tool.
- Save Fortran Environment tool.

If you have existing applications that were created by the CVF COM Server Wizard, you may be able to rebuild them with Intel Visual Fortran. If you need to make changes to the interfaces, you can do so in CVF if you have left it installed. CVF COM Server Wizard projects cannot be converted to Intel® COM Server Wizard projects.

Although Intel Visual Fortran supports all CVF language syntax, incorrect programs which were compilable with CVF may result in error messages when compiled with Intel Visual Fortran. If you encounter unexpected error messages and need assistance resolving them, please contact Intel® Premier Support.

Microsoft Visual Studio Premier Partner Edition*

If you are using Microsoft Visual Studio Premier Partner Edition, the following limitations apply:

- No Microsoft language processors such as Visual C++* or Visual BASIC* are supplied.
- CVF projects cannot be converted to Intel Visual Fortran projects. Instead, create a new Intel Visual Fortran project and add your source files to it.
- The Microsoft Resource Editor, used to create and edit dialog boxes, is not provided. Existing resource files can be used.
- The Intel® C++ Compiler cannot be used in this environment.

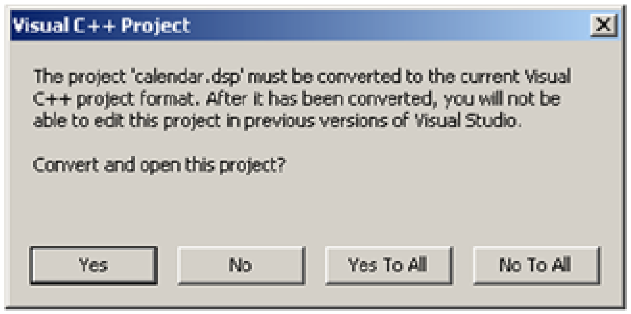
For more information, please refer to the compiler's Release Notes.

Migrating CVF Projects

Intel Visual Fortran includes project-conversion wizards to make it easy to migrate from CVF. Note that the project conversion wizards are not available if you are using Microsoft Visual Studio Premier Partner Edition. The steps and illustrations below assume use of Visual Studio 2005. The process using Visual Studio .NET 2003 is similar.

Conversion is a two-step process:

1. Open the CVF workspace in Visual Studio by right-clicking on the workspace's .DSW file and selecting Open With... Microsoft Visual Studio 2005. You will see a message similar to the following:



Visual C++ Project

The project 'calendar.dsp' must be converted to the current Visual C++ project format. After it has been converted, you will not be able to edit this project in previous versions of Visual Studio.

Convert and open this project?

Figure 1. Initial Project Conversion Dialog

Click Yes To All to convert each project to a Visual C++ project in a "solution" (similar to a workspace).

2. In the right or left pane, you will see the Solution Explorer with the project(s) present. If you do not see the Solution Explorer pane, select View>Solution Explorer. At this point, the conversion to Fortran project(s) is not yet complete. For each project, right-click the project name and select Extract Compaq Visual Fortran Project Items.

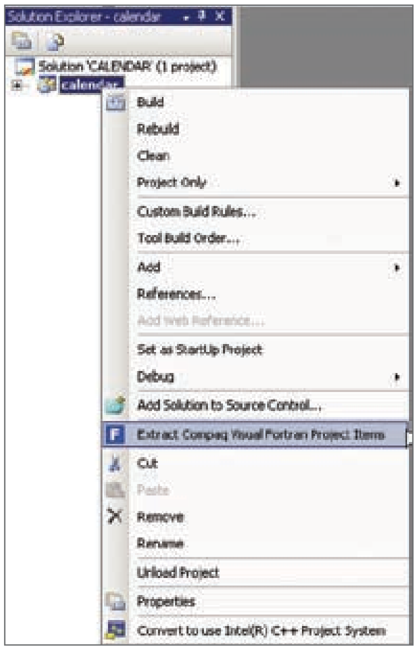


Figure 2. Extract Fortran Project Items Dialog

The project will now be converted.

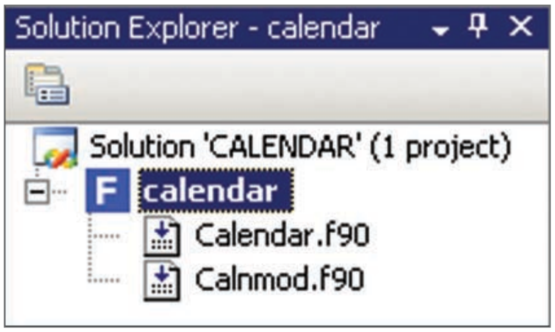
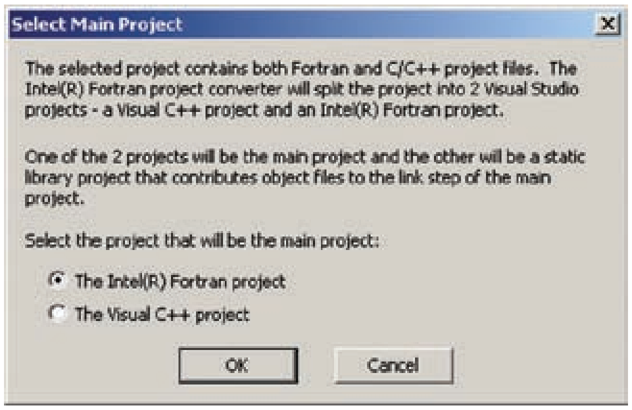


Figure 3. Converted Project

If the CVF project contains both Fortran and C sources, it must be converted into two single-language projects under a solution—one builds a static library and the other links to that library—because Microsoft Visual Studio does not allow multiple languages in a single project. The project conversion wizard asks you which language has the main (linkable) project, Fortran or C, and makes the appropriate adjustments. Figure 4 is an example of converting a mixed-language project.


Select Main Project

The selected project contains both Fortran and C/C++ project files. The Intel(R) Fortran project converter will split the project into 2 Visual Studio projects - a Visual C++ project and an Intel(R) Fortran project.

One of the 2 projects will be the main project and the other will be a static library project that contributes object files to the link step of the main project.

Select the project that will be the main project:

[✓ The Intel(R) Fortran project] The Visual C++ project

Figure 4. Mixed Language Project Dialog Box

After conversion, the mixed-language project looks like Figure 5.

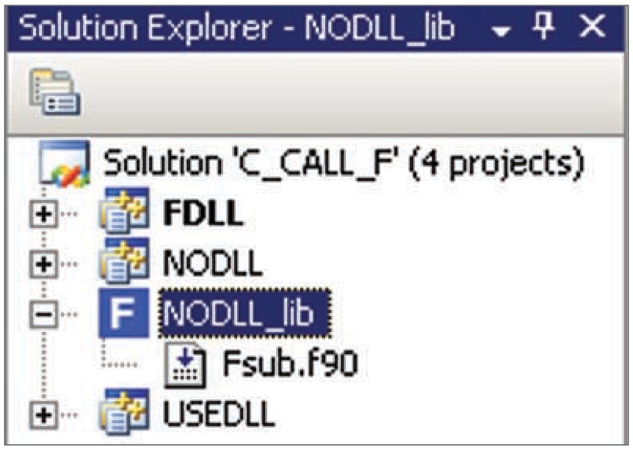

Solution Explorer - NODLL_lib
Solution 'C_CALL_F' (4 projects)
FDLL
NODLL
NODLL_lib
Fsub.f90
USEDLL

Figure 5. Converted Mixed Language Project

In the prior example, the CVF NODLL project was split into a C++ executable project named NODLL and an Intel Fortran static library project NODLL.lib. The conversion wizard automatically makes NODLL_lib a dependant of NODLL, so that the library is built first and then is linked into the C code.

If you convert a CVF project, the conversion wizard will change project settings from the default to enhance compatibility with CVF. An important change is the default calling convention: if it was "Default" in CVF, the conversion wizard changes it to "CVF." (The following section provides more information on calling conventions.) Unless you had a mixed-language application that depended on CVF-specific calling conventions, you should set the default calling convention back to the default, in most cases.

If your mixed-language project was a static library project, two static library projects will be created, one a dependent of the other. Visual Studio will automatically combine the child library with the parent when the solution is built.

Getting Used to the Microsoft Visual Studio 2005 IDE

The Microsoft Visual Studio 2005 Integrated Development Environment (IDE) is so different from the one shared by CVF that some users may find it difficult to perform common tasks. This paper covers some of the major changes, but it is not comprehensive. For more information on using the IDE, see the MSDN Library* documentation that accompanies Microsoft Visual Studio. (If you are using Microsoft Visual Studio Premier Partner Edition, you can access the MSDN Library at http://msdn2.microsoft.com/en-us/library)

Projects and Solutions

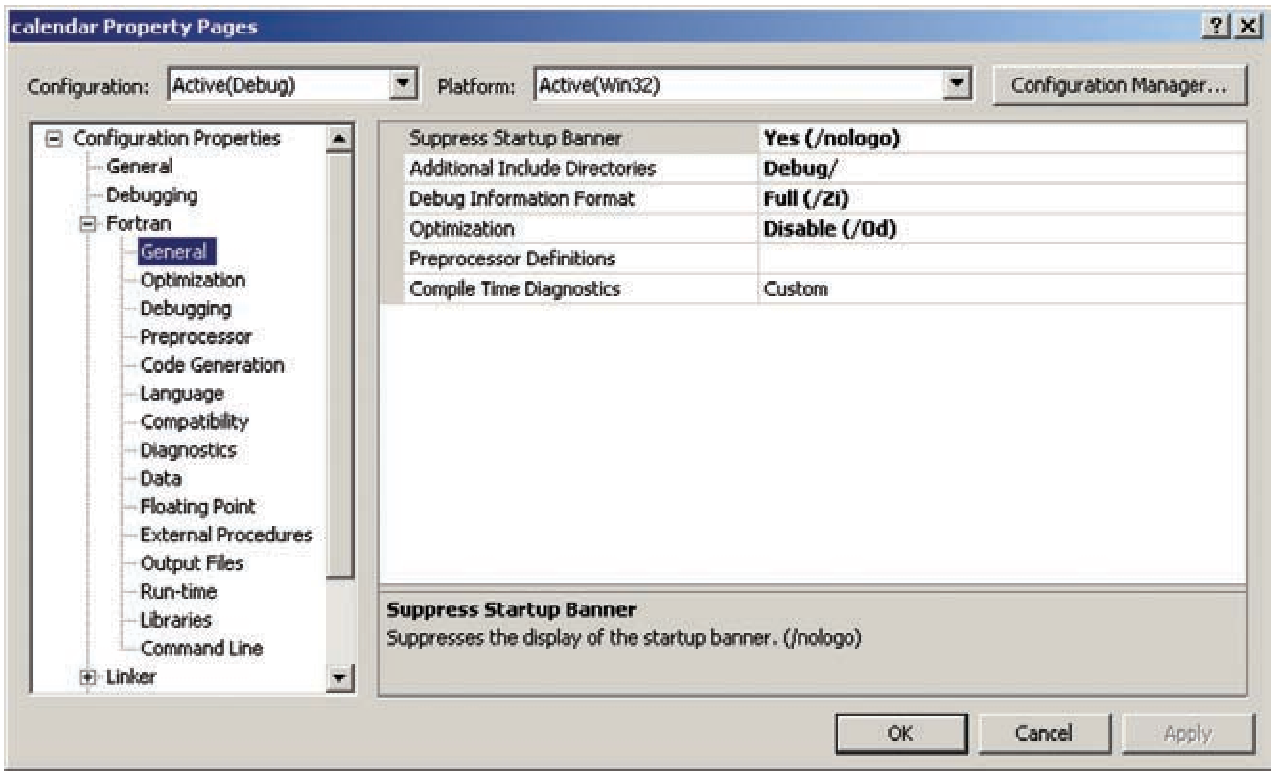
In CVF (and Visual C++ 6.0), "projects" were placed in "workspaces." A CVF workspace was little more than a container for one or more projects and was not involved in the build process. A project built something (an EXE, LIB or DLL file in most cases), and it could contain both Fortran and C code. One project was always designated as "active."

In Visual Studio 2005, projects are substantially the same, but a project can be associated with only one language. For example, if you add C files to a Fortran project, the C files will be ignored. A solution holds multiple projects, but it is different from a workspace because you can build a solution, which builds all of the contained projects in a specified, user-configurable order.

When you have a mixed Fortran and C application, you must put the Fortran code into a Fortran project and the C code into a C (or C++) project. The projects get built separately and then, if appropriate, they are linked together. If the old project was a static library, two static library projects are created, with the objects going into a combined .LIB file. In this case, it does not matter which project you select as being the "main" project.

Changing Settings

Changing settings in the Visual Studio 2005 IDE is different as well. Instead of a tabbed dialog box for Settings, there is a tree-view set of "Property Pages." Figure 6 shows a set of property pages for an example project.


calendar Property Pages

Configuration: Active(Debug)
Platform: Active(Win32)

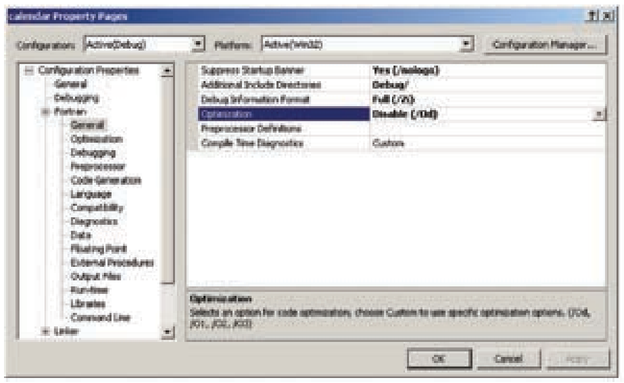
Suppress Startup Banner	**Yes (/nologo)**
Additional Include Directories	**Debug/**
Debug Information Format	**Full (/Zi)**
Optimization	**Disable (/Od)**
Preprocessor Definitions	
Compile Time Diagnostics	Custom

Suppress Startup Banner
Suppresses the display of the startup banner. (/nologo)

Figure 6. Property Pages

In this example, the Fortran General Property Page is displayed. Current property values that are the defaults are shown bolded, and a brief description of the highlighted property is displayed at the bottom of the pane.

1. To change a property value that has specific options, click on the value. An arrow icon displays to the right of the value.



Suppress Startup Banner	**Yes (/nologo)**
Additional Include Directories	**Debug/**
Debug Information Format	**Full (/Zi)**
Optimization	**Disable (/Od)**
Preprocessor Definitions	
Compile Time Diagnostics	Custom

Optimization
Selects an option for code optimization; choose Custom to use specific optimization options. (/Od, /O1, /O2, /O3)

Figure 7. Changing a Property

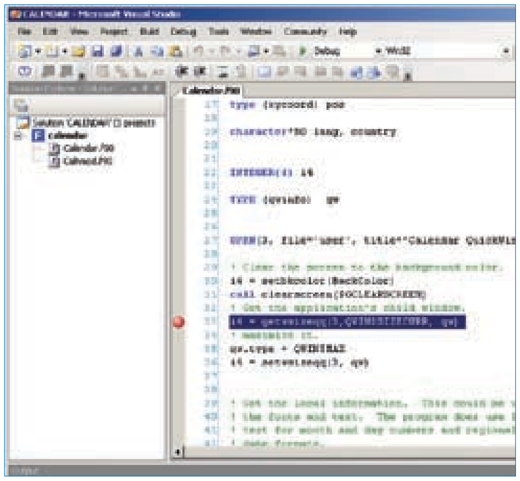
2. Click on the arrow to display the options and select the desired option. Properties that offer a list of items, such as Additional Include Directories in the example above, display a list icon (three dots).

3. Click the icon to open a separate dialog box, where you can enter the values. If you have only one value, type it directly on the Property page. You can directly type properties that are a single text string. In some cases, an arrow icon is available to allow you to select "Inherit from project defaults."

Debugging

Debugging with the Intel Visual Fortran Compiler is similar to using CVF, but some of the controls are in different places.

- To set a breakpoint, click in the left margin next to a statement; the red Stop icon displays.
- To execute under the debugger, click the green triangle "Play" icon next to the configuration name.
- Unlike in CVF, there is not a default button on the toolbar for "Start Without Debugging". You can access this function by selecting it under the Debug menu or by pressing Ctrl+F5. If you want to add the button, select Tools>Customize. In the Commands tab, select Debug under Categories and drag the Start Without Debugging button to the toolbar.



```
type (xycoord) pos

character*80 lang, country


INTEGER(4) i4

TYPE (qwinfo)  qw


OPEN(3, file='user', title='Calendar QuickWin

! Clear the screen to the background color.
i4 = setbkcolor(BackColor)
call clearscreen($GCLEARSCREEN)
! Get the application's child window.
i4 = getwsizeqq(3,QWIN$SIZECURR, qw)
! maximize it.
qw.type = QWIN$MAX
i4 = setwsizeqq(3, qw)


! Get the local information.  This could be
! the fonts and text.  The program does use
! text for month and day numbers and regional
! date formats.
```

Figure 8. Debugging in the Visual Studio Environment

Source Changes

While Intel has taken great care to avoid the need for source changes to permit "rebuild and go," a number of implementation differences between the Intel and Compaq compilers necessitate making changes for selected applications. For the latest information on changes, please refer to the Intel Visual Fortran Compiler Release Notes.

While the part of the compiler that handles Fortran syntax and semantics, often referred to as the "front end," is derived from CVF, improvements have been made in detecting incorrect usage, and you may find that the new compiler issues diagnostic messages for certain usages where the older compiler did not. For example, the compiler now gives an error for a source that makes a call to a non-pure intrinsic such as RANDOM_NUMBER from inside a pure procedure. In such cases, you will need to correct the coding errors. If you believe a diagnostic is inappropriate, please contact Intel® Premier Support.

Default Calling Conventions Have Changed

In CVF, the default calling mechanism was STDCALL, and routine names were 'decorated' by adding @n to the end, where n was the number of bytes of argument list. Intel Visual Fortran Compiler adopts the more common C calling mechanism used by versions 7.1 and earlier of the Intel Fortran Compiler. Routine names are still converted to uppercase by default, and a leading underscore is added, but there is no @n suffix.

Another change is the manner in which CHARACTER argument lengths are passed. In CVF, these were passed immediately following the address of the CHARACTER item, but in Intel Visual Fortran Compiler, all the lengths are passed at the end of the argument list. In other words, the default has changed from /iface:mixed_str_len_arg to /iface:nomixed_str_len_arg.

If you have a Fortran-only application, this change may not be important to you. If, on the other hand, you have a mixed-language application, you need to be aware of the impact of the change in compiler default. You can tell the compiler to use the CVF default on the External Procedures property page, or with the /iface:cvf command-line switch.

A special case where the convention change matters is if your application uses "callback procedures". This is where you pass a routine name from your program as an argument to a library routine. Some kinds of library routines which use callbacks are Win32 API and IMSL libraries, the QSORT routine from the Intel Fortran Portability Library, and also the USEROPEN keyword of the Fortran OPEN statement. Callback routines assume a specific calling convention and if there is a mismatch it can cause stack corruption and unpredictable results.

The Win32 API always uses the STDCALL convention, so if you are passing Fortran routines to Win32 API routines, be sure to add the directive

```
!DEC$ ATTRIBUTES STDCALL,REFERENCE :: routine-name
```

to the callback routine. Most other uses of callbacks, including the IMSL library, assume the compiler's default conventions. If you encounter unexpected run-time errors, try setting the calling convention to "Default" to see if that resolves the issue.

GETARG, IARGC and NARGS Are Now Intrinsic

The command-line inquiry routines GETARG, IARGC and NARGS are now recognized as intrinsic procedures by the compiler. If your application source declares any of these names as EXTERNAL or provides an explicit procedure interface for them, you must remove those declarations to prevent link-time errors.

New Module Names for System and Library Declarations

Compaq Visual Fortran provided modules with definitions of Win32* API routines and symbols, as well as modules for Fortran library routines. These modules had names such as DFWIN, DFLIB, etc. Intel Visual Fortran Compiler provides compatible modules with the same names, but these are wrappers around modules with new names. You do not need to change your sources, but you should begin using the new names in new development. In DFLIB, symbols have been relocated into one of three new modules: IFCORE, IFPORT and IFQWIN. You may find it useful to select the specific module containing the symbols you are interested in.

Old Name	New Name	Description
DFAUTO	IFAUTO	Automation
DFCOM	IFCOM	COM, OLE
DFCOMTY	IFCOMTY	Obsolete; use IFWINTY
DFLIB	IFCORE, IFPORT, IFQWIN	General library, Portability library, QuickWin
DFLOGM	IFLOGM	Dialog
DFLOGMT	IFLOGMT	Dialog types and constants
DFMT	IFMT	Multithread routines
DFNLS	IFNLS	National Language Support
DFOPNGL	IFOPNGL	OpenGL routines
DFOPNGLT	IFOPNGLT	OpenGL types and constants
DFPORT	IFPORT	Portability routines
DFWBASE	IFWBASE	Deprecated WIN16 routines
DFWIN	IFWIN (or individual modules such as KERNEL32, USER32)	Win32* API routines
DFWINA	IFWINA	Renamed Win32 routines which conflict with QuickWin routines
DFWINTY	IFWINTY	Win32 API types and constants

Table 1. Old and New Names of System and Library Modules

The individual WIN32 API modules such as KERNEL32 have the same names in Intel Visual Fortran Compiler as in Compaq Visual Fortran.

Build Changes

In most cases, you will not need to make changes in your build procedures. However, to provide for compatibility with future versions of Intel Fortran compilers, some changes are recommended. This section describes differences that affect building applications.

Compile Command is Now `ifort`

Under CVF, four command names were provided for invoking the compiler: df, f90, f77 and fl32. df and f90 were equivalent, f77 added options for compatibility with Compaq Fortran 77, and fl32 added options for compatibility with Microsoft Fortran PowerStation. In Intel Visual Fortran Compiler, ifort is the preferred command name to invoke the compiler. df is also accepted, but gives a warning that can be suppressed with /quiet. f77 and fl32 are not provided.

If you leave CVF installed on your system, it coexists with the Intel Visual Fortran Compiler. When using the command line, be sure to use the appropriate shortcut in the Start menu to start your command session. For CVF, use "Fortran Command Prompt." For the Intel Visual Fortran Compiler, it is "Fortran Build Environment for applications running on IA-32" Each of these will establish the proper environment for the selected compiler. The IDEs are separate and do not interfere with each other.

Using the IMSL* Fortran Libraries from Visual Numerics

*Intel Visual Fortran Compiler, Professional Edition with IMSL**, includes the IMSL Fortran Library* 6.0 from Visual Numerics, Inc. In addition to many new and updated routines, the new version adds the following features not included in Compaq Visual Fortran Professional Edition:

- Multithreaded libraries for improved performance on multi-core and multiprocessor systems..
- DLL form of the libraries.
- Support for the Intel® 64 (formerly Intel® EM64T) and IA-64 (Intel® Itanium®) processor architectures
- New, unified interface modules making use of Fortran 95 generic interfaces and optional arguments.

Please read the Intel® Visual Fortran Compiler for Windows* Installation Guide for installation instructions.

An existing application that uses the IMSL libraries will need few or no changes, unless you want to take advantage of the new features. You will need to make a change in how the application is built to reference the proper libraries. Below is an overview of this change. See the Intel Visual Fortran Compiler on-disk documentation under Building Applications, Using Libraries for additional details.

A change that may be needed is if your calls to the IMSL library pass callback routines. These are often used to evaluate user functions in "solver" routines. You must make sure that your callback routines use the default calling convention and not CVF's STDCALL. See the above section on the change in calling conventions for more information.

Locating and Referencing the IMSL Libraries

In Compaq Visual Fortran, there was just one set of IMSL libraries: static and non-threaded. In Intel Visual Fortran Compiler, Professional Edition, there are four sets of libraries with different names. Rather than explicitly list the libraries, as was commonly done with CVF, you should instead specify the path to the folder containing the libraries and use one of the supplied INCLUDE files to create a reference to the specific libraries required.

In the Visual Studio IDE, add the path to the IMSL libraries by selecting the menu options Tools > Options > Intel Fortran and inserting in the Project Directories: Libraries list the path to the IMSL libraries (typically `C:\Program Files\VNI\IMSL\FNL600\IA32\LIB`).

Do the same for the IMSL INCLUDE folder, adding it to the list under Project Directories: Includes. The default path for IA-32 is `C:\Program Files\VNI\IMSL\FNL600\IA32\INCLUDE\DLL`.

If you use the command line, the command file that establishes the Intel Visual Fortran environment, `ifortvars.bat`, will automatically establish the IMSL environment if installed.

The remaining step causes the appropriate libraries to be referenced by adding one of the following lines to any one source file in your application:

- `INCLUDE 'link_f90_static.h'`
  – uses the single-processor, static library form of the libraries.
- `INCLUDE 'link_f90_dll.h'`
  – uses the single-processor, dynamic link library form of the libraries.
- `INCLUDE 'link_f90_static_smp.h'`
  – uses the multiprocessor, static library form of the libraries (the DLL form of the Intel® Math Kernel Library will always be used).
- `INCLUDE 'link_f90_dll_smp.h'`
  – uses the multiprocessor, dynamic link library form of the libraries.

For additional options and details, please refer to the Building Applications manualon-disk documentation section on using the IMSL libraries (Building Applications > Using Libraries > Using the IMSL* Mathematical and Statistical Libraries).

Product Features

Intel Visual Fortran Compiler provides a rich feature set that delivers winning performance for both legacy and cutting-edge technologies:

- Full support of Intel multi-core processors and Intel processors supporting Intel EM64T, along with previous Intel processors and architectures.and 64-bit addressing.
- Fortran 2003 language features.
- Quadruple-precision floating point REAL(16) and COMPLEX(32).
- Automatic parallelization.
- OpenMP* support.
- Advanced optimization for new Intel processors.
- Code Coverage and Test Prioritization tools.
- Command Line Debugger (Intel® Debugger).

For more information on these and other features, see the compiler Release Notes.

Getting Help

For "how to" questions, visit the Intel Fortran user forums at http://softwareforums.intel.com/ to ask questions of other knowledgeable users and search for previous discussions that may address your problem.

If you believe you have found a bug in the product, contact Intel® Premier Support. You must first register for support; you are prompted to register when you install the compiler, and you may also register at the Registration Center at http://www.intel.com/software/products/registrationcenter/index.htm. Once registered, log in at https://premier.intel.com/ and submit your support request.

For additional information about Intel compilers and other Intel® Software Development Tools, see the Intel® Software Development Products Web site at http://www.intel.com/software/products/ and the Intel Software Tools Developer Center at http://www.intel.com/cd/ids/developer/asmo-na/eng/dc/tools/index.htm.

For product and purchase information visit:
www.intel.com/software/products