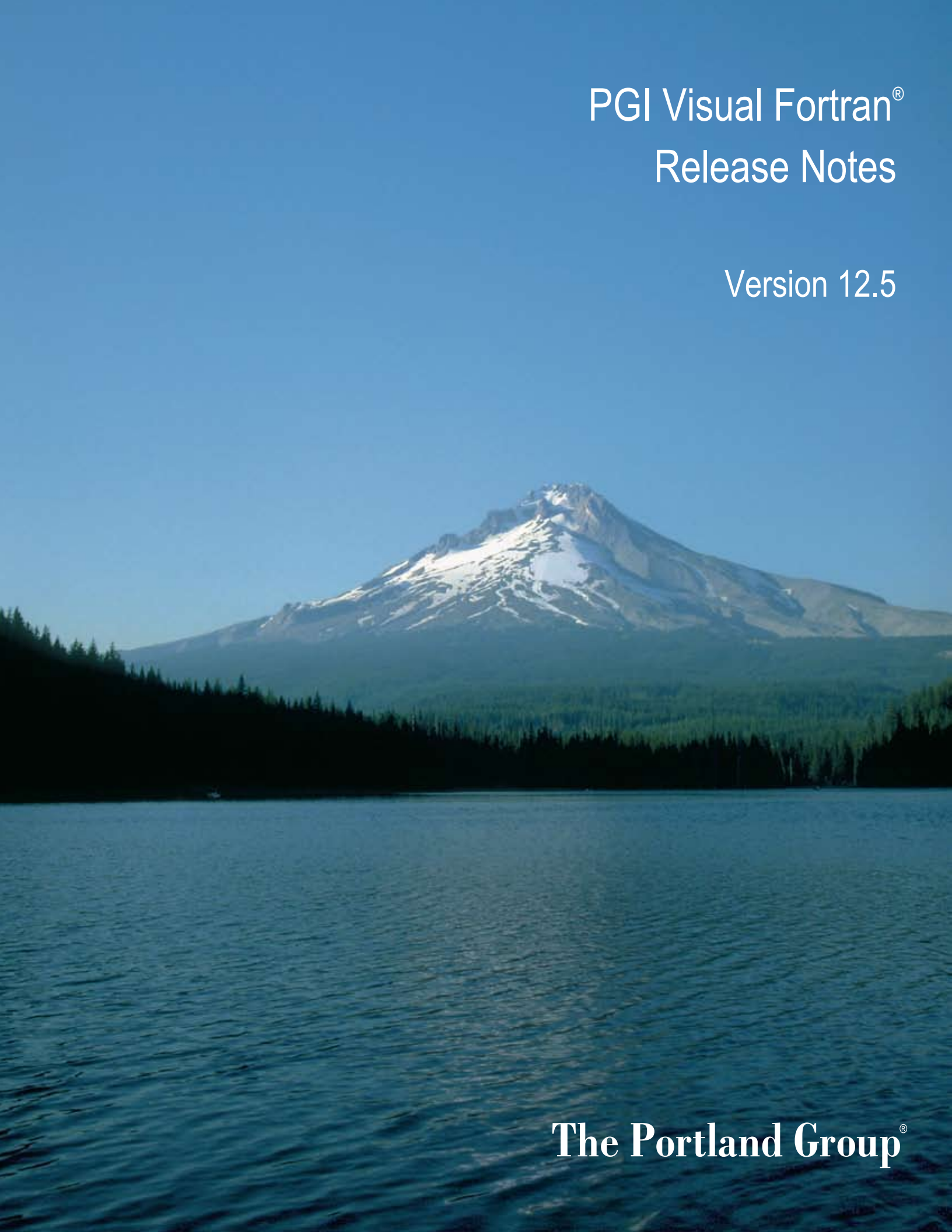
# PGI Visual Fortran®
# Release Notes

Version 12.5



The Portland Group®

PGI Visual Fortran®



Printed in the United States of America

First Printing: Release 2012, version 12.1, January 2012
Second Printing: Release 2012, version 12.2, February 2012
Third Printing: Release 2012, version 12.3, March 2012
Fourth Printing: Release 2012, version 12.4, April 2012
Fourth Printing: Release 2012, version 12.5, May 2012

Technical support: trs@pgroup.com
Sales: sales@pgroup.com
Web: www.pgroup.com


ID: 121301344

# Contents

# Chapter 1. PVF® Release Overview

Welcome to Release 2012 of PGI Visual Fortran®, a set of Fortran compilers and development tools for 32-bit and 64-bit x86-compatible processor-based workstations and servers running versions of the Windows operating system.

This document describes the new features of the PVF IDE interface, differences in the PVF 2012 compilers and tools from previous releases, and late-breaking information not included in the standard product documentation.

PGI Visual Fortran (PVF®) is licensed using FLEXnet, the flexible license management system from Flexera Software*. Instructions for obtaining a permanent license are included in your order confirmation. More information on licensing is in the PVF Installation Guide for this release.

## Product Overview

There are two products in the PVF product family. Each product is integrated with a particular version of Microsoft Visual Studio:

- PGI Visual Fortran 2010

  This product is integrated with Microsoft Visual Studio 2010 (VS 2010).

- PGI Visual Fortran 2008

  This product is integrated with Microsoft Visual Studio 2008 (VS 2008).

Throughout this document, "PGI Visual Fortran" and “PVF” refer to all PVF products collectively. Similarly, "Microsoft Visual Studio" refers to VS 2010 and VS 2008. When it is necessary to distinguish between the products, the document uses the full product name.

Single-user node-locked and multi-user network floating license options are available for all PVF products. When a node-locked license is used, one user at a time can use PVF on the single system where it is installed. When a network floating license is used, a system is selected as the server and it controls the licensing, and users from any of the client machines connected to the server can use PVF. Thus multiple users can simultaneously use PVF, up to the maximum number of users allowed by the license.

PVF provides a complete Fortran development environment fully integrated with Microsoft Visual Studio. It includes a custom Fortran Build Engine that automatically derives build dependencies, a Fortran-aware editor,

a custom PGI Debug Engine integrated with Visual Studio, PGI Fortran compilers, and PVF-specific property pages to control the configuration of all of these.

Release 2012 of PGI Visual Fortran includes the following components:

- PGFORTRAN™ native OpenMP and auto-parallelizing Fortran 90/95/2003 compiler.
- PGF77® native OpenMP and auto-parallelizing FORTRAN 77 compiler.
- PGPROF® Performance Profiler
- PVF Visual Studio integration components
- AMD Core Math Library 4.4.0 (ACML)
- PVF documentation

PGI Visual Fortran is available with the Microsoft Visual Studio Shell. Use this package if you do not already have Visual Studio installed on your system. Otherwise, download the versions of PVF without the VS Shell, since these are much smaller.

## Terms and Definitions

These release notes contain a number of terms and definitions with which you may or may not be familiar. If you encounter a term in these notes with which you are not familiar, please refer to the online glossary at

www.pgroup.com/support/definitions.htm

These two terms are used throughout the documentation to reflect groups of processors:

- AMD64 – a 64-bit processor from AMD designed to be binary compatible with 32-bit x86 processors, and incorporating new features such as additional registers and 64-bit addressing support for improved performance and greatly increased memory range. This term includes the AMD Athlon64, AMD Opteron, AMD Turion, AMD Barcelona, AMD Shanghai, AMD Istanbul, and AMD Bulldozer processors.
- Intel 64 – a 64-bit IA32 processor with Extended Memory 64-bit Technology extensions designed to be binary compatible with AMD64 processors. This includes Intel Pentium 4, Intel Xeon, Intel Core 2, Intel Core 2 Duo (Penryn), Intel Core (i3, i5, i7) both first generation (Nehalem) and second generation (Sandy Bridge) processors.

# Chapter 2. New or Modified Features

This chapter contains the new or modified features of this release of PGI Visual Fortran as compared to prior releases.

## What's New in PVF Release 2012

### 12.5 Updates and Additions

- Type parameters for derived types, also known as parameterized derived types, are now supported.

### Updates and Additions Prior to 12.5

- `dflib` now contains the routine `makedirqq` that allows users to create a new directory.
- OpenMP nested parallelism support
- PGI Accelerator x64+GPU native Fortran 2003 and C99 compilers and CUDA Fortran now support the CUDA 4.0 Toolkit as the default toolkit

## New or Modified Compiler Options

Unknown options are treated as errors instead of warnings. This feature means it is a compiler error to pass switches that are not known to the compiler; however, you can use the switch `-noswitcherror` to issue warnings instead of errors for unknown switches.

## New or Modified Runtime Library Routines

PGI 2012 supports new runtime library routines associated with the PGI Accelerator compilers. For more information, refer to the "Using an Accelerator" chapter of the PGI Visual Fortran User's Guide.

## PGI Accelerator and CUDA Fortran Enhancements

PGI Accelerator x64+GPU native Fortran 95/03 and C99 compilers and CUDA Fortran now support the CUDA 4.0 Toolkit. PGI compilers and tools continue to support previous versions of CUDA provided these toolkits exist on your system from a previous installation.

The default toolkit for 12.1 is CUDA 4.0. Thus, CUDA Fortran host programs should be recompiled, as the PGI 12.1 CUDA Fortran runtime libraries are not compatible with previous CUDA Fortran releases.

To specify the version of the **CUDA Toolkit** that is targeted by the compilers, use one of the following properties:

## For PGI Accelerator model:

Use the property: `Fortran | Target Accelerators | NVIDIA: CUDA Toolkit`

When Target NVIDIA Accelerator is set to `Yes`, you can specify the version of the NVIDIA CUDA Toolkit targeted by the compilers.

`Default`: The compiler selects the default CUDA Toolkit version.
`4.0`: Specifies use of toolkit version 4.0.
`3.2`: Specifies use of toolkit version 3.2.
`3.1`: Specifies use of toolkit version 3.1.
`3.0`: Specifies use of toolkit version 3.0.
`2.3`: Specifies use of toolkit version 2.3.

Selecting one of these properties is equivalent to adding the associated switch to the PVF compilation and link lines:

```
-ta=nvidia[:cuda2.3 | cuda3.0 | cuda3.1 | cuda3.2 | cuda4.0]
```

## For CUDA Fortran:

Use the property: `Fortran | Language | CUDA Fortran Toolkit`

When Enable CUDA Fortran is set to `Yes`, you can specify the version of the CUDA Toolkit targeted by the compilers.

`Default`: The compiler selects the default CUDA Toolkit version.
`4.0`: Specifies use of toolkit version 4.0.
`3.2`: Specifies use of toolkit version 3.2.
`3.1`: Specifies use of toolkit version 3.1.
`3.0`: Specifies use of toolkit version 3.0.
`2.3`: Specifies use of toolkit version 2.3.

Selecting one of these properties is equivalent to adding the associated switch to the PVF compilation and link lines:

```
-Mcuda[=cuda2.3 | cuda3.0 | cuda3.1 | cuda3.2 | cuda4.0]
```

## PGI Accelerator Runtime Routines

For a complete description of the PGI Accelerator model runtime routines available in version 12.5, refer to Chapter 4, "PGI Accelerator Compilers Reference" of the *PGI Compiler Reference Manual*.

## Memory Management in CUDA

A new memory management routine, `cudaMemGetInfo`, returns the amount of free and total memory available (in bytes) for allocation on the device.

The syntax for `cudaMemGetInfo` is:

```
integer function cudaMemGetInfo( free, total )
     integer(kind=cuda_count_kind) :: free, total
```

## Declaring Interfaces to CUDA Device Built-in Routines

A Fortran module is available to declare interfaces to many of the CUDA device built-in routines.

To access this module, add this line to your Fortran program:

```
use cudadevice
```

You can also use these routines in CUDA Fortran global and device subprograms, in CUF kernels, and in PGI Accelerator compute regions both in Fortran and in C. Further, the PGI compilers come with implementations of these routines for host code, though these implementations are not specifically optimized for the host.

For a complete list of the CUDA built-in routines that are available, refer to the *PGI CUDA Fortran Programming and Reference*.

# Chapter 3. Selecting an Alternate Compiler

Each release of PGI Visual Fortran contains two components - the newest release of PVF and the newest release of the PGI compilers and tools that PVF targets.

When PVF is installed onto a system that contains a previous version of PVF, the previous version of PVF is replaced. The previous version of the PGI compilers and tools, however, remains installed side-by-side with the new version of the PGI compilers and tools. By default, the new version of PVF will use the new version of the compilers and tools. Previous versions of the compilers and tools may be uninstalled using Control Panel | Add or Remove Programs.

There are two ways to use previous versions of the compilers:

- Use a different compiler release for a single project.
- Use a different compiler release for all projects.

The method to use depends on the situation.

## For a Single Project

To use a different compiler release for a single project, you use the compiler flag -V<ver> to target the compiler with version <ver>. This method is the recommended way to target a different compiler release.

For example, -V10.1 causes the compiler driver to invoke the 10.1 version of the PGI compilers if these are installed.

To use this option within a PVF project, add it to the Additional options section of the Fortran | Command Line and Linker | Command Line property pages.

## For All Projects

You can use a different compiler release for all projects.

The Tools | Options dialog within PVF contains entries that can be changed to use a previous version of the PGI compilers. Under Projects and Solutions | PVF Directories, there are entries for Executable Directories, Include and Module Directories, and Library Directories.

- For the x64 platform, each of these entries includes a line containing $(PGIToolsDir). To change the compilers used for the x64 platform, change each of the lines containing $(PGIToolsDir) to contain the path to the desired bin, include, and lib directories.
- For the 32-bit Windows platform, these entries include a line containing $(PGIToolsDir) on 32-bit Windows systems or $(PGIToolsDir32) on 64-bit Windows systems. To change the compilers used for the 32-bit Windows platform, change each of the lines containing $(PGIToolsDir) or $(PGIToolsDir32) to contain the path to the desired bin, include, and lib directories.

### Warning

The debug engine in PVF 2012 is not compatible with previous releases. If you use Tools | Options to target a release prior to 2012, you cannot use PVF to debug. Instead, use the `-V` method described earlier in this chapter to select an alternate compiler.

# Chapter 4. Distribution and Deployment

Once you have successfully built, debugged and tuned your application, you may want to distribute it to users who need to run it on a variety of systems. This chapter addresses how to effectively distribute applications built using PGI compilers and tools.

## Application Deployment and Redistributables

Programs built with PGI compilers may depend on runtime library files. These library files must be distributed with such programs to enable them to execute on systems where the PGI compilers are not installed. There are PGI redistributable files for all platforms. On Windows, PGI also supplies Microsoft redistributable files.

### PGI Redistributables

PGI Visual Fortran includes redistributable directories which contain all of the PGI dynamically linked libraries that can be re-distributed by PVF 2012 licensees under the terms of the PGI End-User License Agreement (EULA). For reference, a copy of the PGI EULA in PDF form is included in the release.

The following paths for the redistributable directories assume 'C:' is the system drive.

- On a 32-bit Windows system, the redistributable directory is:

```
C:\Program Files\PGI\win32\12.5\REDIST
```

- On a 64-bit Windows system, there are two redistributable directories:

```
C:\Program Files\PGI\win64\12.5\REDIST
C:\Program Files (x86)\PGI\win32\12.5\REDIST
```

The redistributable directories contain the PGI runtime library DLLs for all supported targets. This enables users of the PGI compilers to create packages of executables and PGI runtime libraries that execute successfully on almost any PGI-supported target system, subject to the requirement that end-users of the executable have properly initialized their environment to use the relevant version of the PGI DLLs.

## Microsoft Redistributables

PGI Visual Fortran includes Microsoft Open Tools, the essential tools and libraries required to compile, link, and execute programs on Windows. PVF 2012 includes the latest version, version 10, of the Microsoft Open Tools.

The Microsoft Open Tools directory contains a subdirectory named REDIST. PGI 2012 licensees may redistribute the files contained in this directory in accordance with the terms of the associated license agreements.

### Note

On Windows, runtime libraries built for debugging (e.g. `msvcrtd` and `libcmtd`) are not included with PGI Visual Fortran. When a program is linked with `-g` for debugging, the standard non-debug versions of both the PGI runtime libraries and the Microsoft runtime libraries are always used. This limitation does not affect debugging of application code.

# Chapter 5. Troubleshooting Tips and Known Limitations

This chapter contains information about known limitations, documentation errors, and corrections that have occurred to PVF 2012. Whenever possible, a workaround is provided.

For up-to-date information about the state of the current release, visit the frequently asked questions (FAQ) section on pgroup.com at: www.pgroup.com/support/index.htm

## Use MPI in PVF Limitations

- The multi-process debug style known as "Run One At a Time" is not supported in this release.

## PVF IDE Limitations

The issues in this section are related to IDE limitations.

- Integration with source code revision control systems is not supported.
- When moving a project from one drive to another, all .d files for the project should be deleted and the whole project should be rebuilt. When moving a solution from one system to another, also delete the solution's Visual Studio Solution User Options file (.suo).
- The Resources property pages are limited. Use the Resources | Command Line property page to pass arguments to the resource compiler. Resource compiler output must be placed in the intermediate directory for build dependency checking to work properly on resource files.
- There are several properties that take paths or pathnames as values. In general, these may not work as expected if they are set to the project directory $(ProjectDir) or if they are empty, unless empty is the default. Specifically:

  *General* | *Output Directory* should not be empty or set to $(ProjectDir).
  *General* | *Intermediate Directory* should not be empty or set to $(ProjectDir).
  *Fortran* | *Output* | *Object File Name* should not be empty or set to $(ProjectDir).
  *Fortran* | *Output* | *Module Path* should not be empty or set to include $(ProjectDir).

- Dragging and dropping files in the Solution Explorer that are currently open in the Editor may result in a file becoming "orphaned." Close files before attempting to drag-and-drop them.

## PVF Debugging Limitations

The following limitations apply to PVF debugging:

- Debugging of unified binaries is not fully supported. The names of some subprograms are modified in the creation of the unified binary, and the PVF debug engine does not translate these names back to the names used in the application source code. For more information on debugging a unified binary, see www.pgroup.com/support/tools.htm.
- In some situations, using the Watch window may be unreliable for local variables. Calling a function or subroutine from within the scope of the watched local variable may cause missed events and/or false positive events. Local variables may be watched reliably if program scope does not leave the scope of the watched variable.
- Rolling over Fortran arrays during a debug session is not supported when Visual Studio is in Hex mode. This limitation also affects Watch and Quick Watch windows.

  *Workaround*: deselect Hex mode when rolling over arrays.

## PGI Compiler Limitations

The frequently asked questions (FAQ) section on the pgroup.com web page at www.pgroup.com/support/index.htm provides more up to date information about the state of the current release.

- Take extra care when using `-Mprof` with PVF runtime library DLLs. To build an executable for profiling, use of the static libraries is recommended. The static libraries are used by default in the absence of `-Bdynamic`.
- Using `-Mpfi` and `-mp` together is not supported. The `-Mpfi` flag disables `-mp` at compile time, which can cause runtime errors in programs that depend on interpretation of OpenMP directives or pragmas. Programs that do not depend on OpenMP processing for correctness can still use profile feedback. Using the `-Mpfo` flag does not disable OpenMP processing.
- The `-i8` option can make programs incompatible with the ACML library; use of any INTEGER*8 array size argument can cause failures with these libraries. Visit developer.amd.com to check for compatible ACML libraries.
- ACML 4.4.0 is built using the `-fastsse` compile/link option, which includes `-Mcache_align`. When linking with ACML on 32-bit Windows, all program units must be compiled with `-Mcache_align`, or an aggregate option such as `-fastsse`, which incorporates `-Mcache_align`. This process is not an issue on 64-bit targets where the stack is 16-byte aligned by default. You can use the lower-performance, but fully portable, `blas` and `lapack` libraries on CPUs that do not support SSE instructions.

## Corrections

Refer to www.pgroup.com/support/release_tprs.htm for a complete, up-to-date table of technical problem reports, TPRs, fixed in recent releases of the PGI compilers and tools. The table contains a summary description of each problem as well as the version in which it was fixed.

# Chapter 6. Contact Information

You can contact The Portland Group at:

The Portland Group
STMicroelectronics, Inc.
Two Centerpointe Drive, Suite 320
Lake Oswego, OR 97035 USA

The PGI User Forum is monitored by members of the PGI engineering and support teams as well as other PGI customers. The forum newsgroups may contain answers to commonly asked questions. Log in to the PGI website to access the forum:

www.pgroup.com/userforum/index.php

Or contact us electronically using any of the following means:

| | |
|---|---|
| Fax | +1-503-682-2637 |
| Sales | sales@pgroup.com |
| Support | trs@pgroup.com |
| WWW | www.pgroup.com |

All technical support is by email or submissions using an online form at www.pgroup.com/support. Phone support is not currently available.

Many questions and problems can be resolved at our frequently asked questions (FAQ) site at www.pgroup.com/support/faq.htm.

PGI documentation is available at www.pgroup.com/resources/docs.htm or in your local copy of the documentation in the release directory doc/index.htm.