Document number: 304970-005US

# 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.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See http://www.intel.com/products/processor\_number for details. This document contains information on products in the design phase of development.

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino Atom, Centrino Atom Inside, Centrino Inside, Centrino Iogo, Core Inside, FlashFile, i960, InstantIP, Intel, Intel Iogo, Intel386, Intel486, IntelDX2, IntelDX4, IntelSX2, Intel Atom, Intel Atom Inside, Intel Core, Intel Inside, Intel Inside Iogo, Intel. Leap ahead., Intel. Leap ahead. Iogo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Viiv, Intel vPro, Intel XScale, Itanium, Itanium Inside, MCS, MMX, Oplus, OverDrive, PDCharm, Pentium, Pentium Inside, skoool, Sound Mark, The Journey Inside, Viiv Inside, vPro Inside, VTune, Xeon, and Xeon Inside 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 (C) 1996-2008, Intel Corporation. All rights reserved.

Portions Copyright (C) 2001, Hewlett-Packard Development Company, L.P.

# **Table of Contents**

Intel(R) Fortran Compiler User and Reference Guides	1
Intel® Fortran Compiler User and Reference Guides	1
Disclaimer and Legal Information	1
Welcome to the Intel® Fortran Compiler	2
See Also	2
Conventions	3
Notational Conventions	3
Platform Labels	7
Introduction to the Intel® Fortran Compiler	8
Product Website and Support	8
System Requirements	9
FLEXIm* Electronic Licensing	9
Related Publications	9
Tutorial information	9
Standard and Specification Documents	10
Associated Intel Documents	11
Optimization and Vectorization Terminology and Technology	11
Additional Training on the Intel® Fortran Compiler	12
Compiler Options	13
Overview: Compiler Options	13
New Options	14
Deprecated and Removed Compiler Options	34
Alphabetical Compiler Options	40
Quick Reference Guides and Cross References	699
Related Options	795
Floating-point Operations	805
Overview: Floating-point Operations	805
Floating-point Options Quick Reference	806
Understanding Floating-point Operations	811

Tuning Performance	822
Handling Floating-point Exceptions	826
Understanding IEEE Floating-point Standard	847

Intel® Fortran Compiler User and Reference Guides

Document number: 304970-005US

**Start Here** 

www.intel.com

<u>Disclaimer and Legal Information</u>

# 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.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See http://www.intel.com/products/processor\_number for details. This document contains information on products in the design phase of development.

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino Atom, Centrino Atom Inside, Centrino Inside, Centrino Iogo, Core Inside, FlashFile, i960, InstantIP, Intel, Intel Iogo, Intel386, Intel486, IntelDX2, IntelDX4, IntelSX2, Intel Atom, Intel Atom Inside, Intel Core, Intel Inside, Intel Inside Iogo, Intel. Leap ahead., Intel. Leap ahead. Iogo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Viiv, Intel vPro, Intel XScale, Itanium, Itanium Inside, MCS, MMX, Oplus, OverDrive, PDCharm, Pentium, Pentium Inside, skoool, Sound Mark, The Journey Inside, Viiv Inside, vPro Inside, VTune, Xeon, and Xeon Inside 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 (C) 1996-2008, Intel Corporation. All rights reserved.

Portions Copyright (C) 2001, Hewlett-Packard Development Company, L.P.

# Welcome to the Intel® Fortran Compiler

The Intel® Fortran Compiler lets you build and optimize Fortran applications for the Linux\* OS (operating system).

#### See Also

- Introduction
- Building Applications
- Compiler Options
- Optimizing Applications
- Floating-point Operations
- Language Reference

For details on getting started with the Intel Fortran Compiler, see:

- Getting Started
- Invoking the Compiler from the Command Line

#### Conventions

Information in this documentation applies to all supported operating systems and architectures unless otherwise specified.

This documentation uses the following conventions:

- Notational Conventions
- Platform Labels

Conventions

THIS TYPE	Indicates statements,	data types,	directives, and	other language

keywords. Examples of statement keywords are WRITE,

INTEGER, DO, and OPEN.

this type Indicates command-line or option arguments, new terms, or

emphasized text. Most new terms are defined in the Glossary.

This type Indicates a code example.

This type Indicates what you type as input.

**This type** Indicates menu names, menu items, button names, dialog

window names, and other user-interface items.

**File>Open** Menu names and menu items joined by a greater than (>) sign

indicate a sequence of actions. For example, "Click File>Open"

indicates that in the **File** menu, click **Open** to perform this

action.

{value | value} Indicates a choice of items or values. You can usually only

choose one of the values in the braces.

[item] Indicates items that are optional. Brackets are also used in code

examples to show arrays.

item [, item ]... Indicates that the item preceding the ellipsis (three dots) can be

repeated. In some code examples, a horizontal ellipsis means

that not all of the statements are shown.

Windows\* OS These terms refer to all supported Microsoft\* Windows\*

Windows operating systems.

operating

system

Linux\* OS These terms refer to all supported Linux\* operating systems.

Linux operating

system

Mac OS\* X These terms refer to Intel®-based systems running the Mac

Mac OS X OS\* X operating system.

operating

system

Microsoft An asterisk at the end of a word or name indicates it is a third-

Windows XP\* party product trademark.

compiler option This term refers to Windows\* OS options, Linux\* OS options, or

MAC OS\* X options that can be used on the compiler command

line.

#### Conventions Used in Compiler Options

/option or A slash before an option name indicates the option is

-option available on Windows OS. A dash before an option name

indicates the option is available on Linux OS\* and Mac

OS\* X systems. For example:

Windows option: /fast

Linux and Mac OS X option: -fast

Note: If an option is available on Windows\* OS, Linux\* OS, and Mac OS\* X systems, no slash or dash appears in the general description of the option. The slash and dash will only appear where the option syntax is described.

/option:argument Indicates that an option requires a argument (parameter).

or

For example, you must specify an argument for the

-option argument following options:

Windows OS option: /Qdiag-error-limit:n

Linux OS and Mac OS X option: -diag-error-limit *n* 

/option: keyword

Indicates that an option requires one of the *keyword* 

or

or

values.

-option *keyword* 

/option[:keyword] Indicates that the option can be used alone or with an

optional keyword.

-option [keyword]

Indicates that the option can be used alone or with an option[n]

optional value; for example, in /Qunroll[:n] or -

unroll[n], the n can be omitted or a valid value can be

specified for n.

option[-] Indicates that a trailing hyphen disables the option; for

example, /Qglobal\_hoist-disables the Windows OS

option /Qglobal\_hoist.

[no]option or

Indicates that "no" or "no-" preceding an option disables

the option. For example: [no-]option

In the Windows OS option / [no] traceback,

/traceback enables the option, while /notraceback

disables it.

In the Linux OS and Mac OS X option - [no-

]global\_hoist, -global\_hoist enables the option,

while -no-global\_hoist disables it.

In some options, the "no" appears later in the option name; for example, -fno-alias disables the -falias

option.

#### Conventions Used in Language Reference

This color Indicates extensions to the Fortran 95 Standard. These extensions may or may not be implemented by other compilers that conform to the language standard.

Intel Fortran This term refers to the name of the common compiler language supported by the Intel® Visual Fortran Compiler and the Intel® Fortran Compiler. For more information on these compilers, see <a href="http://developer.intel.com/software/products/">http://developer.intel.com/software/products/</a>.

Fortran This term refers to language information that is common to ANSI FORTRAN 77, ANSI/ISO Fortran 95 and 90, and Intel Fortran.

Fortran This term refers to language information that is common to ANSI 95/90 FORTRAN 77, ANSI/ISO Fortran 95, ANSI/ISO Fortran 90, and Intel Fortran.

Fortran 95 This term refers to language features specific to ANSI/ISO Fortran 95.

This term refers to the INTEGER(KIND=1), INTEGER(KIND=2), INTEGER (INTEGER(KIND=4)), and INTEGER(KIND=8) data types as a group.

real This term refers to the REAL (REAL(KIND=4)), DOUBLE PRECISION (REAL(KIND=8)), and REAL(KIND=16) data types as a group.

REAL This term refers to the default data type of objects declared to be REAL. REAL is equivalent to REAL(KIND=4), unless a compiler option specifies otherwise.

complex This term refers to the COMPLEX (COMPLEX(KIND=4)), DOUBLE COMPLEX (COMPLEX(KIND=8)), and COMPLEX(KIND=16) data types as a group.

Iogical This term refers to the LOGICAL(KIND=1), LOGICAL(KIND=2), LOGICAL (LOGICAL(KIND=4)), and LOGICAL(KIND=8) data types as a group.

Compatibility This term introduces a list of the projects or libraries that are compatible with the library routine.

< Tab> This symbol indicates a nonprinting tab character.

^ This symbol indicates a nonprinting blank character.

# **Platform Labels**

A platform is a combination of operating system and central processing unit (CPU) that provides a distinct environment in which to use a product (in this case, a computer language). An example of a platform is Microsoft\* Windows\* XP on processors using IA-32 architecture.

In this documentation, information applies to all supported platforms unless it is otherwise labeled for a specific platform (or platforms).

These labels may be used to identify specific platforms:

- L\*X Applies to Linux\* OS on processors using IA-32 architecture, Intel® 64 architecture, and IA-64 architecture.
- L\*X32 Applies to Linux\* OS on processors using IA-32 architecture and Intel® 64 architecture.
- L\*X64 Applies to Linux OS on processors using IA-64 architecture.
- M\*X Applies to Apple\* Mac OS\* X on processors using IA-32 architecture and Intel® 64 architecture.

- M\*X32 Applies to Apple\* Mac OS\* X on processors using IA-32 architecture.
- M\*X64 Applies to Apple\* Mac OS\* X on processors using Intel® 64 architecture.
- W\*32 Applies to Microsoft Windows\* 2000, Windows XP, and Windows Server 2003 on processors using IA-32 architecture and Intel® 64 architecture. For a complete list of supported Windows\* operating systems, see your Release Notes.
- W\*64 Applies to Microsoft Windows\* XP operating systems on IA-64 architecture.
- i32 Applies to 32-bit operating systems on IA-32 architecture.
- i64em Applies to 32-bit operating systems on Intel® 64 architecture.
- i64 Applies to 64-bit operating systems on IA-64 architecture.

# Introduction to the Intel® Fortran Compiler

The Intel® Fortran Compiler can generate code for IA-32, Intel® 64, or IA-64 applications on any Intel®-based Linux\* system. IA-32 applications (32-bit) can run on all Intel®-based Linux systems. Intel® 64 applications and IA-64 applications can only run on Intel® 64-based or IA-64-based Linux systems. For more information about the compiler features and other components, see your *Release Notes*.

This documentation assumes that you are familiar with the Fortran programming language and with your processor's architecture. You should also be familiar with the host computer's operating system.

#### **Product Website and Support**

For general information on support for Intel software products, visit the Intel web site <a href="http://developer.intel.com/software/products/">http://developer.intel.com/software/products/</a>

At this site, you will find comprehensive product information, including:

- Links to each product, where you will find technical information such as white papers and articles
- Links to user forums

#### Links to news and events

To find technical support information, to register your product, or to contact Intel, please visit: <a href="http://www.intel.com/software/products/support/">http://www.intel.com/software/products/support/</a>

For additional information, see the Technical Support section of your Release Notes.

## **System Requirements**

For detailed information on system requirements, see the Release Notes.

#### **FLEXIm\* Electronic Licensing**

The Intel® Fortran Compiler uses Macrovision\*'s FLEXIm\* licensing technology.

The compiler requires a valid license file in the *licenses* directory in the installation path. The default directory is */opt/intel/licenses*.

License files have a file extension of .lic.

For information on how to install and use the Intel® License Manager for FLEXIm to configure a license server for systems using counted licenses, see *Using the Intel® License Manager for FLEXIm\** (flex\_ug.pdf).

#### Related Publications

#### **Tutorial information**

The following commercially published documents provide reference or tutorial information on Fortran 2003, Fortran 95, and Fortran 90:

- Fortran 95/2003 for Scientists & Engineers by S. Chapman; published by McGraw-Hill Science/Engineering/Math, ISBN 0073191574.
- Fortran 95/2003 Explained by M. Metcalf, J. Reid, and M. Cohen; published by Oxford University Press, ISBN 0-19-852693-8.
- Compaq Visual Fortran by N. Lawrence; published by Digital Press\*
   (Butterworth-Heinemann), ISBN 1-55558-249-4.
- Digital Visual Fortran Programmers Guide by M. Etzel and K. Dickinson;
   published by Digital Press (Butterworth-Heinemann), ISBN 1-55558-218-4
- Fortran 90 Explained by M. Metcalf and J. Reid; published by Oxford University Press, ISBN 0-19-853772-7.
- Fortran 90/95 Explained by M. Metcalf and J. Reid; published by Oxford University Press, ISBN 0-19-851888-9.

- Fortran 90/95 for Scientists and Engineers by S. Chapman; published by McGraw-Hill, ISBN 0-07-011938-4.
- Fortran 90 Handbook by J. Adams, W. Brainerd, J. Martin, B. Smith, and J. Wagener; published by Intertext Publications (McGraw-Hill), ISBN 0-07-000406-4.
- Fortran 90 Programming by T. Ellis, I. Philips, and T. Lahey; published by Addison-Wesley, ISBN 0201-54446-6.
- Introduction to Fortran 90/95 by Stephen J. Chapman; published by McGraw-Hill, ISBN 0-07-011969-4.
- User's guide to Fortran 90, Second Edition by W. Brainerd, C. Goldberg, and
   J. Adams; published by Unicomp, ISBN 0-07-000248-7.

Intel does not endorse these books or recommend them over other books on the same subjects.

#### **Standard and Specification Documents**

The following copyrighted standard and specification documents provide descriptions of many of the features found in Intel® Fortran:

- American National Standard Programming Language FORTRAN, ANSI X3.9-1978
- American National Standard Programming Language Fortran 90, ANSI X3.198-1992
  - This Standard is equivalent to: International Standards Organization Programming Language Fortran, ISO/IEC 1539:1991 (E).
- American National Standard Programming Language Fortran 95, ANSI X3J3/96-007
  - This Standard is equivalent to: International Standards Organization Programming Language Fortran, ISO/IEC 1539-1:1997 (E).
- International Standards Organization Information Technology Programming Languages - Fortran, ISO/IEC 1539-1:2004 (E)
   This is the Fortran 2003 Standard.
- High Performance Fortran Language Specification, Version 1.1, Technical Report CRPC-TR-92225

- OpenMP Fortran Application Program Interface, Version 1.1, November 1999
- OpenMP Fortran Application Program Interface, Version 2.0, November 2000

#### **Associated Intel Documents**

The following Intel documents provide additional information about the Intel® Fortran Compiler, Intel® architecture, Intel® processors, or tools:

- Using the Intel® License Manager for FLEXIm\*
- Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 1:
   Basic Architecture, Intel Corporation
- Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 2A:
   Instruction Set Reference, A-M, Intel Corporation
- Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 2B:
   Instruction Set Reference, N-Z, Intel Corporation
- Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3A:
   System Programming Guide, Intel Corporation
- Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3B:
   System Programming Guide, Intel Corporation
- Intel® 64 and IA-32 Architectures Optimization Reference Manual
- Intel® Itanium® Architecture Software Developer's Manual Volume 1:
   Application Architecture, Revision 2.2
- Intel® Itanium® Architecture Software Developer's Manual Volume 2:
   System Architecture, Revision 2.2
- Intel® Itanium® Architecture Software Developer's Manual Volume 3:
   Instruction Set Reference, Revision 2.2
- Intel® Processor Identification with the CPUID Instruction, Intel Corporation, doc. number 241618
- IA-64 Architecture Assembler User's Guide
- IA-64 Architecture Assembly Language Reference Guide

Most Intel documents can be found at the Intel web site

http://developer.intel.com/software/products/

#### **Optimization and Vectorization Terminology and Technology**

The following documents provide details on basic optimization and vectorization terminology and technology:

- Intel® Architecture Optimization Reference Manual
- Dependence Analysis, Utpal Banerjee (A Book Series on Loop Transformations for Restructuring Compilers). Kluwer Academic Publishers. 1997.
- The Structure of Computers and Computation: Volume I, David J. Kuck. John Wiley and Sons, New York, 1978.
- Loop Transformations for Restructuring Compilers: The Foundations, Utpal Banerjee (A Book Series on Loop Transformations for Restructuring Compilers). Kluwer Academic Publishers. 1993.
- Loop parallelization, Utpal Banerjee (A Book Series on Loop Transformations for Restructuring Compilers). Kluwer Academic Publishers. 1994.
- High Performance Compilers for Parallel Computers, Michael J. Wolfe.
   Addison-Wesley, Redwood City. 1996.
- Supercompilers for Parallel and Vector Computers, H. Zima. ACM Press, New York, 1990.
- An Auto-vectorizing Compiler for the Intel® Architecture, Aart Bik, Paul Grey,
   Milind Girkar, and Xinmin Tian. Submitted for publication
- Efficient Exploitation of Parallelism on Pentium® III and Pentium® 4
   Processor-Based Systems, Aart Bik, Milind Girkar, Paul Grey, and Xinmin Tian.
- The Software Vectorization Handbook. Applying Multimedia Extensions for Maximum Performance, A.J.C. Bik. Intel Press, June, 2004.
- Multi-Core Programming: Increasing Performance through Software
   Multithreading, Shameem Akhter and Jason Roberts. Intel Press, April, 2006.

#### Additional Training on the Intel® Fortran Compiler

For additional training on the Intel Fortran Compiler, choose a course in the Intel® Software College - Course Catalog at <a href="http://shale.intel.com/SoftwareCollege/CourseCatalog.asp">http://shale.intel.com/SoftwareCollege/CourseCatalog.asp</a>

For additional technical product information including white papers about Intel compilers, open the page associated with your product at <a href="http://developer.intel.com/software/products/">http://developer.intel.com/software/products/</a>

# **Compiler Options**

#### **Overview: Compiler Options**

This document provides details on all current Linux\*, Mac OS\* X, and Windows\* compiler options.

It provides the following information:

#### New options

This topic lists new compiler options in this release.

#### Deprecated

This topic lists deprecated and removed compiler options for this release. Some deprecated options show suggested replacement options.

#### Alphabetical Compiler Options

This topic is the main source in the documentation set for general information on all compiler options. Options are described in alphabetical order. The <a href="Overview">Overview</a> describes what information appears in each compiler option description.

#### Quick Reference Guide and Cross Reference

This topic contains tables summarizing compiler options. The tables show the option name, a short description of the option, the default setting for the option, and the equivalent option on the operating system, if any.

#### Related Options

This topic lists related options that can be used under certain conditions.

In this guide, compiler options are available on all supported operating systems and architectures unless otherwise identified.

For further information on compiler options, see Building Applications and Optimizing Applications.

#### **Functional Groupings of Compiler Options**

To see functional groupings of compiler options, specify a functional category for option help on the command line. For example, to see a list of options that affect diagnostic messages displayed by the compiler, enter one of the following commands:

```
-help diagnostics ! Linux and Mac OS X systems
/help diagnostics ! Windows systems
```

For details on the categories you can specify, see <u>help</u>.

## **New Options**

This topic lists the options that provide new functionality in this release.

Some compiler options are only available on certain systems, as indicated by these labels:

	Label	Meaning		
i	32	The option is available on systems using IA-32 architecture.		
i	64em	The option is available on systems using Intel® 64 architecture.		
i	64	The option is available on systems using IA-64 architecture.		
If no label appears, the option is available on all supported systems.				
	If "only" appears in the label, the option is only available on the identified system.			

section. For information on conventions used in this table, see  $\underline{\text{Conventions}}$ .

New compiler options are listed in tables below:

• The first table lists new options that are available on Windows\* systems.

For more details on the options, refer to the Alphabetical Compiler Options

The second table lists new options that are available on Linux\* and Mac OS\*
 X systems. If an option is only available on one of these operating systems, it is labeled.

Windows* Options	Description	Default
/arch:IA32	Generates code that	OFF
(i32 only)	will run on any	
	Pentium or later	

Windows* Options	Description	Default
	processor.	
/arch:SSE3 (i32, i64em)	Optimizes for Intel® Streaming SIMD Extensions 3 (Intel® SSE3).	OFF
/arch:SSSE3 (i32, i64em)	Optimizes for Intel® Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3).	OFF
/arch:SSE4.1 (i32, i64em)	Optimizes for Intel® Streaming SIMD Extensions 4 Vectorizing Compiler and Media Accelerators.	OFF
/GS (i32, i64em)	Determines whether the compiler generates code that detects some buffer overruns.	/GS-
/QaxSSE2 (i32, i64em)	Can generate Intel® SSE2 and SSE instructions for Intel processors, and it can optimize for Intel® Pentium® 4 processors, Intel®	OFF

Windows* Options	Description	Default
	Pentium® M processors, and Intel® Xeon® processors with Intel® SSE2.	
/QaxSSE3 (i32, i64em)	Can generate Intel® SSE3, SSE2, and SSE instructions for Intel processors and it can optimize for processors based on Intel® Core™ microarchitecture and Intel NetBurst® microarchitecture.	OFF
/QaxSSSE3 (i32, i64em)	Can generate Intel® SSSE3, SSE3, SSE2, and SSE instructions for Intel processors and it can optimize for the Intel® Core™2 Duo processor family.	OFF
/QaxSSE4.1 (i32, i64em)	Can generate Intel® SSE4 Vectorizing Compiler and Media Accelerator instructions for Intel processors. Can	OFF

Windows* Options	Description	Default
	generate Intel® SSSE3, SSE3, SSE2, and SSE instructions and it can optimize for Intel® 45nm Hi-k next generation Intel® Core™ microarchitecture.	
/QaxSSE4.2 (i32, i64em)	Can generate Intel® SSE4 Efficient Accelerated String and Text Processing instructions supported by Intel® Core™ i7 processors. Can generate Intel® SSE4 Vectorizing Compiler and Media Accelerator, Intel® SSSE3, SSE3, SSE2, and SSE instructions and it can optimize for the Intel® Core™ processor family.	OFF
/Qdiag-error-limit:n	Specifies the maximum number of errors allowed before compilation stops.	n=30

Windows* Options	Description	Default
/Qdiag-once:id[,id,]	Tells the compiler to issue one or more diagnostic messages only once.	OFF
/Qfast-transcendentals	Enables the compiler to replace calls to transcendental functions with faster but less precise implementations.	OFF
/Qfma (i64 only)	Enables the combining of floating-point multiplies and add/subtract operations.	ON
/Qfp-relaxed (i64 only)	Enables use of faster but slightly less accurate code sequences for math functions.	OFF
/Qinstruction:[no]movbe(i32, i64em)	Determines whether MOVBE instructions are generated for Intel processors.	ON
/Qopenmp-link:library	Controls whether the compiler links to static or dynamic OpenMP run-time libraries.	

Windows* Options	Description	Default
/Qopenmp- threadprivate:type	Lets you specify an OpenMP* threadprivate implementation.	/Qopenmp- threadprivate:legacy
/Qopt-block-factor:n	Lets you specify a loop blocking factor.	OFF
/Qopt-jump- tables:keyword	Enables or disables generation of jump tables for switch statements.	/Qopt-jump- tables:default
/Qopt-loadpair (i64 only)	Enables loadpair optimization.	/Qopt-loadpair-
/Qopt-mod-versioning (i64 only)	Enables versioning of modulo operations for certain types of operands.	
/Qopt-prefetch-initial-values (i64 only)	Enables or disables prefetches that are issued before a loop is entered.	/Qopt-prefetch- initial-values
/Qopt-prefetch-issue- excl-hint (i64 only)	Determines whether the compiler issues prefetches for stores with exclusive hint.	/Qopt-prefetch- issue-excl-hint-
/Qopt-prefetch-next- iteration (i64 only)	Enables or disables prefetches for a memory access in the	/Qopt-prefetch-next- iteration

Windows* Options	Description	Default
	next iteration of a loop.	
/Qopt-subscript-in- range (i32, i64em)	Determines whether the compiler assumes no overflows in the intermediate computation of subscript expressions in loops.	
/Qprof-data-order	Enables or disables data ordering if profiling information is enabled.	/Qprof-data-order
/Qprof-func-order	Enables or disables function ordering if profiling information is enabled.	/Qprof-func-order
/Qprof-hotness- threshold	Lets you set the hotness threshold for function grouping and function ordering.	OFF
/Qprof-src-dir	Determines whether directory information of the source file under compilation is considered when looking up profile data records.	/Qprof-src-dir

Windows* Options	Description	Default
/Qprof-src-root	Lets you use relative directory paths when looking up profile data and specifies a directory as the base.	OFF
/Qprof-src-root-cwd	Lets you use relative directory paths when looking up profile data and specifies the current working directory as the base.	OFF
/Qtcollect-filter	Lets you enable or disable the instrumentation of specified functions.	OFF
/Quse-msasm-symbols (i32, i64em)	Tells the compiler to use a dollar sign ("\$") when producing symbol names.	OFF
/Qvc9 (i32, i64em)	Specifies compatibility with Microsoft* Visual Studio 2008.	varies
/Qvec (i32, i64em)	Enables or disables vectorization and transformations enabled for vectorization.	/Qvec

Windows* Options	Description	Default
/QxHost (i32, i64em)	Can generate specialized code paths for the highest instruction set and processor available on the compilation host.	OFF
/QxSSE2 (i32, i64em)	Can generate Intel® SSE2 and SSE instructions for Intel processors, and it can optimize for Intel® Pentium® 4 processors, Intel® Pentium® M processors, and Intel® Xeon® processors with Intel® SSE2.	
/QxSSE3 (i32, i64em)	Can generate Intel® SSE3, SSE2, and SSE instructions for Intel processors, and it can optimize for processors based on Intel® Core™ microarchitecture and Intel NetBurst® microarchitecture.	OFF

Windows* Options	Description	Default
/QxSSE3_ATOM (i32, i64em)	Can generate  MOVBE instructions for Intel processors and it can optimize for the Intel® Atom™ processor and Intel® Centrino® Atom™ Processor Technology.	OFF
/QxSSSE3 (i32, i64em)	Can generate Intel® SSSE3, SSE3, SSE2, and SSE instructions for Intel processors and it can optimize for the Intel® Core™2 Duo processor family.	OFF
/QxSSE4.1 (i32, i64em)	Can generate Intel® SSE4 Vectorizing Compiler and Media Accelerator instructions for Intel processors. Can generate Intel® SSSE3, SSE3, SSE2, and SSE instructions and it can optimize for Intel® 45nm Hi-k next generation Intel® Core™	OFF

Windows* Options	Description	Default
	microarchitecture.	
/QxSSE4.2 (i32, i64em)	Can generate Intel® SSE4 Efficient Accelerated String and Text Processing instructions supported by Intel® Core™ i7 processors. Can generate Intel® SSE4 Vectorizing Compiler and Media Accelerator, Intel® SSSE3, SSE3, SSE2, and SSE instructions and it can optimize for the Intel® Core™ processor family.	OFF
Linux* and Mac OS* X Options	Description	Default
-axSSE2 (i32, i64em)	Can generate Intel® SSE2 and SSE instructions for Intel processors, and it can optimize for Intel® Pentium® 4 processors, Intel® Pentium® M processors, and Intel®	OFF

Linux* and Mac OS* X Options	Description	Default
	Xeon® processors with Intel® SSE2.	
-axSSE3 (i32, i64em)	Can generate Intel® SSE3, SSE2, and SSE instructions for Intel processors, and it can optimize for processors based on Intel® Core microarchitecture and Intel NetBurst® microarchitecture.	OFF
-axSSSE3 (i32, i64em)	Can generate Intel® SSSE3, SSE3, SSE2, and SSE instructions for Intel processors and it can optimize for the Intel® Core™2 Duo processor family.	OFF
-axSSE4.1 (i32, i64em)	Can generate Intel® SSE4 Vectorizing Compiler and Media Accelerator instructions for Intel processors. Can generate Intel® SSSE3, SSE3, SSE2, and SSE instructions and it can optimize for	OFF

Linux* and Mac OS* X Options	Description	Default
	Intel® 45nm Hi-k next generation Intel® Core™ microarchitecture.	
-axSSE4.2 (i32, i64em)	Can generate Intel® SSE4 Efficient Accelerated String and Text Processing instructions supported by Intel® Core™ i7 processors. Can generate Intel® SSE4 Vectorizing Compiler and Media Accelerator, Intel® SSSE3, SSE3, SSE2, and SSE instructions and it can optimize for the Intel® Core™ processor family.	OFF
-diag-error-limit n	Specifies the maximum number of errors allowed before compilation stops.	n=30
-diag-once id[,id,]	Tells the compiler to issue one or more diagnostic messages	OFF

Linux* and Mac OS* X Options	Description	Default
	only once.	
-falign-stack (i32 only)	Tells the compiler the stack alignment to use on entry to routines.	
-fast-transcendentals	Enables the compiler to replace calls to transcendental functions with faster but less precise implementation.	
-finline	Tells the compiler to inline functions declared with cDEC\$ ATTRIBUTES FORCEINLINE.	-fno-inline
-fma (i64 only; Linux only)	Enables the combining of floating-point multiplies and add/subtract operations.	ON
-fp-relaxed (i64 only; Linux only)	Enables use of faster but slightly less accurate code sequences for math functions.	OFF
-fpie	Tells the compiler to	OFF
(Linux only)		

Linux* and Mac OS* X Options	Description	Default
	generate position- independent code to link into executables.	
-fstack-protector (i32, i64em)	Determines whether the compiler generates code that detects some buffer overruns. Same as option -fstack-security-check.	-fno-stack-protector
-m32, -m64 (i32, i64em)	Tells the compiler to generate code for a specific architecture.	OFF
-mia32 (i32 only)	Generates code that will run on any Pentium or later processor.	OFF
- minstruction=[no]movbe (i32, i64em)	Determines whether MOVBE instructions are generated for Intel processors.	ON
-mssse3 (i32, i64em)	Generates code for Intel® Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3).	Linux systems: OFF Mac OS X systems using Intel® 64 architecture: ON
-msse4.1 (i32, i64em)	Generates code for Intel® Streaming SIMD	OFF

Linux* and Mac OS* X Options	Description	Default
	Extensions 4 Vectorizing Compiler and Media Accelerators.	
-openmp-link library	Controls whether the compiler links to static or dynamic OpenMP run-time libraries.	-openmp-link dynamic
-openmp-	Lets you specify an	-openmp-
threadprivate=type	OpenMP* threadprivate	threadprivate=legacy
(Linux only)	implementation.	
-opt-block-factor=n	Lets you specify a loop blocking factor.	OFF
-opt-jump-	Enables or disables	-opt-jump-
tables=keyword	generation of jump tables for switch statements.	tables=default
-opt-loadpair (i64 only; Linux only)	Enables loadpair optimization.	-no-opt-loadpair
-opt-mod-versioning	Enables versioning of	-no-opt-mod-
(i64 only; Linux only)	modulo operations for	versioning
	certain types of operands.	
-opt-prefetch-initial-	Enables or disables	-opt-prefetch-
values	prefetches that are	initial-values
(i64 only; Linux only)	issued before a loop is	

Linux* and Mac OS* X Options	Description	Default
	entered.	
-opt-prefetch-issue- excl-hint (i64 only)	Determines whether the compiler issues prefetches for stores with exclusive hint.	-no-opt-prefetch- issue-excl-hint
-opt-prefetch-next- iteration (i64 only; Linux only)	Enables or disables prefetches for a memory access in the next iteration of a loop.	-opt-prefetch-next- iteration
-opt-subscript-in- range (i32, i64em)	Determines whether the compiler assumes no overflows in the intermediate computation of subscript expressions in loops.	-no-opt-subscript- in-range
-pie (Linux only)	Produces a position- independent executable on processors that support it.	OFF
-prof-data-order (Linux only)	Enables or disables data ordering if profiling information is enabled.	-no-prof-data-order
-prof-func-groups (i32, i64em; Linux only)	Enables or disables function grouping if	-no-prof-func-groups

Linux* and Mac OS* X Options	Description	Default
	profiling information is enabled.	
-prof-func-order (Linux only)	Enables or disables function ordering if profiling information is enabled.	-no-prof-func-order
-prof-hotness- threshold (Linux only)	Lets you set the hotness threshold for function grouping and function ordering.	OFF
-prof-src-root	Lets you use relative directory paths when looking up profile data and specifies a directory as the base.	OFF
-prof-src-root-cwd	Lets you use relative directory paths when looking up profile data and specifies the current working directory as the base.	OFF
-staticlib (i32, i64em; Mac OS X only)	Invokes the libtool command to generate static libraries.	OFF
-tcollect-filter (Linux only)	Lets you enable or disable the	OFF

Linux* and Mac OS* X Options	Description	Default
	instrumentation of specified functions.	
-vec (i32, i64em)	Enables or disables vectorization and transformations enabled for vectorization.	-vec
-xHost (i32, i64em)	Can generate instructions for the highest instruction set and processor available on the compilation host.	OFF
-xsse2 (i32, i64em)	Can generate Intel® SSE2 and SSE instructions for Intel processors, and it can optimize for Intel® Pentium® 4 processors, Intel® Pentium® M processors, and Intel® Xeon® processors with Intel® SSE2.	Linux systems:ON Mac OS X systems: OFF
-xSSE3 (i32, i64em)	Can generate Intel® SSE3, SSE2, and SSE instructions for Intel	Linux systems:OFF Mac OS X systems using IA-32 architecture: ON

Linux* and Mac OS* X Options	Description	Default
	processors and it can optimize for processors based on Intel® Core™ microarchitecture and Intel NetBurst® microarchitecture.	
-xSSE3_ATOM (i32, i64em)	Can generate MOVBE instructions for Intel processors and it can optimize for the Intel® Atom™ processor and Intel® Centrino® Atom™ Processor Technology.	OFF
-xSSSE3 (i32, i64em)	Can generate Intel® SSSE3, SSE3, SSE2, and SSE instructions for Intel processors and it can optimize for the Intel® Core™2 Duo processor family.	Linux systems:OFF Mac OS X systems using Intel® 64 architecture: ON
-xSSE4.1 (i32, i64em)	Can generate Intel® SSE4 Vectorizing Compiler and Media Accelerator instructions for Intel processors. Can generate Intel®	OFF

Linux* and Mac OS* X Options	Description	Default
	SSSE3, SSE3, SSE2, and SSE instructions and it can optimize for Intel® 45nm Hi-k next generation Intel® Core™ microarchitecture.	
-xsse4.2 (i32, i64em)	Can generate Intel®  SSE4 Efficient  Accelerated String and Text Processing instructions supported by Intel® Core™ i7 processors. Can generate Intel® SSE4  Vectorizing Compiler and Media Accelerator, Intel® SSSE3, SSE3, SSE2, and SSE instructions and it can optimize for the Intel®  Core™ processor family.	OFF

## **Deprecated and Removed Compiler Options**

This topic lists deprecated and removed compiler options and suggests replacement options, if any are available.

# **Deprecated Options**

Occasionally, compiler options are marked as "deprecated." Deprecated options are still supported in the current release, but are planned to be unsupported in future releases.

The following options are deprecated in this release of the compiler:

Linux* and Mac OS* X Options	Suggested Replacement
-axK	None
-axN	-axSSE2
-axP	-axSSE3
-axS	-axSSE4.1
-axT	-axSSSE3
-axW	-msse2
-func-groups	-prof-func-groups
-i-dynamic	-shared-intel
-i-static	-static-intel
-inline-debug-info	-debug
-IPF-flt-eval-method0	-fp-model source
-IPF-fltacc	-fp-model precise
-no-IPF-fltacc	-fp-model fast
-IPF-fma	-fma
-IPF-fp-relaxed	-fp-relaxed
-march=pentiumii	None
-march=pentiumiii	-march=pentium3
-тсри	-mtune
-mp	-fp-model
-0b	-inline-level

# Linux\* and Mac OS\* X Options Suggested Replacement

-xT -xW	-xSSSE3 -msse2
-xS	-xSSE4.1
-xP	-xSSE3
-x0	-msse3
-xN	-xSSE2
-xK	-mia32
-use-pch	-pch-use
-use-asm	None
-prof-genx	-prof-gen=srcpos
-prefetch	-opt-prefetch
-openmpS	-openmp-stubs
-openmpP	-openmp
-openmp-lib legacy	None

Windows* Options	Suggested Replacement
/4Nb	/check:none
/4Yb	/check:all
/debug:partial	None
/Fm	/map
/G5	None
/G6 (or /GB)	None
/G7	None
/Ge	/Gs0

Windows* Options	Suggested Replacement
/ML and/MLd	None
/Op	/fp
/QaxK	None
/QaxN	/QaxSSE2
/QaxP	/QaxSSE3
/QaxS	/QaxSSE4.1
/QaxT	/QaxSSSE3
/QaxW	/arch:SSE2
/Qinline-debug-info	None
/QIPF-flt-eval-method(	)/fp:source
/QIPF-fltacc	/fp:precise
/QIPF-fltacc-	/fp:fast
/QIPF-fma	/Qfma
/QIPF-fp-relaxed	/Qfp-relaxed
/Qopenmp-lib:legacy	None
/Qprefetch	/Qopt-prefetch
/Qprof-genx	/Qprof-gen=srcpos
/Quse-asm	None
/QxK	None
/QxN	/QxSSE2
/Qx0	/arch:SSE3
/QxP	/QxSSE3
/QxS	/QxSSE4.1

Windows* Options	Suggested Replacement
/QxT	/QxSSSE3
/QxW	/arch:SSE2
/Zd	/debug:minimal

Deprecated options are not limited to this list.

## **Removed Options**

Some compiler options are no longer supported and have been removed. If you use one of these options, the compiler issues a warning, ignores the option, and then proceeds with compilation.

This version of the compiler no longer supports the following compiler options:

Linux* and Mac OS* X Options	Suggested Replacement
-axB	-axSSE2
-axi	None
-axM	None
-cxxlib-gcc[=dir]	-cxxlib[=dir]
-cxxlib-icc	None
-F	-preprocess-only or -P
-fp	-fno-omit-frame-pointer
-fpstkchk	-fp-stack-check
-IPF-fp-speculation	-fp-speculation
-ipo-obj (and -ipo_obj)	None
-Kpic, -KPIC	-fpic
-mtune=itanium	None
-nobss-init	-no-bss-init

Linux* and Mac OS* X Options	Suggested Replacement
-opt-report-level	-opt-report
-prof-format-32	None
-prof-gen-sampling	None
-db	-p
-shared-libcxa	-shared-libgcc
-ssp	None
-static-libcxa	-static-libgcc
-syntax	-syntax-only or -fsyntax-only
-tpp1	None
-tpp2	-mtune=itanium2
-tpp5	None
-tpp6	None
-tpp7	-mtune=pentium4
-xB	-xSSE2
-xi	None
-xM	None

Windows* Options	Suggested Replacement
/4ccD (and /4ccd)	None
/G1	None
/QaxB	/QaxSSE2
/Qaxi	None
/QaxM	None
/Qfpstkchk	/Qfp-stack-check

Windows* Options	Suggested Replacement
/QIPF-fp-speculation	/Qfp-speculation
/Qipo-obj (and /Qipo_obj	) None
/Qopt-report-level	/Qopt-report
/Qprof-format-32	None
/Qprof-gen-sampling	None
/Qssp	None
/Qvc6	None
/Qvc7	None
/QxB	/QxSSE2
/Qxi	None
/QxM	None

Removed options are not limited to these lists.

## **Alphabetical Compiler Options**

## **Compiler Option Descriptions and General Rules**

This section describes all the current Linux\*, Mac OS\* X, and Windows\* compiler options in alphabetical order.

## **Option Descriptions**

Each option description contains the following information:

- A short description of the option.
- IDE Equivalent

This shows information related to the integrated development environment (IDE) Property Pages on Windows, Linux, and Mac OS X systems. It shows on which Property Page the option appears, and under what category it's listed. The Windows IDE is Microsoft\* Visual Studio\* .NET; the Linux IDE is Eclipse\*; the Mac OS X IDE is Xcode\*. If the option has no IDE equivalent, it

will specify "None". Note that in this release, there is no IDE support for Fortran on Linux.

#### Architectures

This shows the architectures where the option is valid. Possible architectures are:

- IA-32 architecture
- Intel® 64 architecture
- IA-64 architecture

### Syntax

This shows the syntax on Linux and Mac OS X systems and the syntax on Windows systems. If the option has no syntax on one of these systems, that is, the option is not valid on a particular system, it will specify "None".

## Arguments

This shows any arguments (parameters) that are related to the option. If the option has no arguments, it will specify "None".

#### Default

This shows the default setting for the option.

#### Description

This shows the full description of the option. It may also include further information on any applicable arguments.

## Alternate Options

These are options that are synonyms of the described option. If there are no alternate options, it will specify "None".

Many options have an older spelling where underscores ("\_") instead of hyphens ("-") connect the main option names. The older spelling is a valid alternate option name.

Some option descriptions may also have the following:

## Example

This shows a short example that includes the option

See Also

This shows where you can get further information on the option or related options.

## **General Rules for Compiler Options**

You cannot combine options with a single dash (Linux and Mac OS X) or slash (Windows). For example:

- On Linux and Mac OS X systems: This is incorrect: -wc; this is correct: -w -
- $\bullet$  On Windows systems: This is incorrect: /wc; this is correct: /w /c All Linux and Mac OS X compiler options are case sensitive. Many Windows options are case sensitive. Some options have different meanings depending on

their case; for example, option "c" prevents linking, but option "C" checks for certain conditions at run time.

Options specified on the command line apply to all files named on the command line.

Options can take arguments in the form of file names, strings, letters, or numbers. If a string includes spaces, the string must be enclosed in quotation marks. For example:

- On Linux and Mac OS X systems, -dynamic-linker mylink (file name) or -Umacro3 (string)
- On Windows systems, /Famyfile.s (file name) or /V"version 5.0" (string) Compiler options can appear in any order.

On Windows systems, all compiler options must precede /link options, if any, on the command line.

Unless you specify certain options, the command line will both compile and link the files you specify.

You can abbreviate some option names, entering as many characters as are needed to uniquely identify the option.

Certain options accept one or more keyword arguments following the option name. For example, the arch option accepts several keywords.

To specify multiple keywords, you typically specify the option multiple times. However, there are exceptions; for example, the following are valid: -axNB (Linux) or /QaxNB (Windows).

## Note

On Windows systems, you can sometimes use a comma to separate keywords. For example, the following is valid:

ifort /warn:usage,declarations test.f90

On these systems, you can use an equals sign (=) instead of the colon:

ifort /warn=usage,declarations test.f90

Compiler options remain in effect for the whole compilation unless overridden by a compiler directive.

To disable an option, specify the negative form of the option.

On Windows systems, you can also disable one or more options by specifying option /od last on the command line.



On Windows systems, the /od option is part of a mutually-exclusive group of options that includes /od, /o1, /o2, /o3, and /ox. The last of any of these options specified on the command line will override the previous options from this group.

If there are enabling and disabling versions of an option on the command line, the last one on the command line takes precedence.

## **Lists and Functional Groupings of Compiler Options**

To see a list of all the compiler options, specify option help on the command line. To see functional groupings of compiler options, specify a functional category for option help. For example, to see a list of options that affect diagnostic messages displayed by the compiler, enter one of the following commands:

```
-help diagnostics ! Linux and Mac OS X systems
/help diagnostics ! Windows systems
For details on the categories you can specify, see help.
1
See onetrip.
412, 414, 418
See integer-size.
4L72, 4L80, 4L132
See extend-source.
4Na, 4Ya
See automatic.
4Naltparam, 4Yaltparam
See altparam.
4Nb,4Yb
See check.
4Nd,4Yd
See warn.
4Nf
See <u>fixed</u>.
4Nportlib, 4Yportlib
Determines whether the compiler links to the library of portability routines.
IDE Equivalent
```

Windows: Libraries > Use Portlib Library

Linux: None

Mac OS X: None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: None

Windows: /4Nportlib

/4Yportlib

## Arguments

None

#### Default

/4Yportlib The library of portability routines is linked during compilation.

## Description

Option /4Yportlib causes the compiler to link to the library of portability routines. This also includes Intel's functions for Microsoft\* compatibility.

Option /4Nportlib prevents the compiler from linking to the library of portability routines.

## **Alternate Options**

None

#### See Also

Building Applications: Portability Routines

## 4Ns,4Ys

See stand. 4R8,4R16 See real-size. 4Yf See free. 4Nportlib, 4Yportlib Determines whether the compiler links to the library of portability routines. **IDE Equivalent** Windows: Libraries > Use Portlib Library Linux: None Mac OS X: None Architectures IA-32, Intel® 64, IA-64 architectures **Syntax** Linux and Mac OS X: None Windows: /4Nportlib /4Yportlib Arguments None Default /4Yportlib The library of portability routines is linked during compilation.

Description

Option /4Yportlib causes the compiler to link to the library of portability routines. This also includes Intel's functions for Microsoft\* compatibility.

Option /4Nportlib prevents the compiler from linking to the library of portability routines.

#### **Alternate Options**

None

#### See Also

Building Applications: Portability Routines

66

See <u>f66</u>.

## 72,80,132

See extend-source.

## align

Tells the compiler how to align certain data items.

## **IDE Equivalent**

Windows: Data > Structure Member Alignment (/align:recnbyte)

Data > Common Element Alignment (/align:[no]commons,

/align:[no]dcommons)

Data > SEQUENCE Types Obey Alignment Rules (/align:[no]sequence)

Linux: None

Mac OS X: Data > Structure Member Alignment (-align

rec<1,2,4,8,16>byte)

Data > Common Element Alignment (-align [no]commons,

/align:[no]dcommons)

Data > SEQUENCE Types Obey Alignment Rules (-align [no]sequence)

## Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -align [keyword]

-noalign

Windows: /align[:keyword]

/noalign

### Arguments

keyword Specifies the data items to align. Possible values are:

none Prevents padding bytes anywhere in

common blocks and structures.

[no]commons Affects alignment of common block

entities.

[no]dcommons Affects alignment of common block

entities.

[no]records Affects alignment of derived-type

components and fields of record

structures.

rechbyte Specifies a size boundary for derived-

type components and fields of record

structures.

[no]sequence Affects alignment of sequenced

derived-type components.

all Adds padding bytes whenever

possible to data items in common

blocks and structures.

## Default

nocommons	Adds no padding bytes for alignment of common blocks.
nodcommmons	Adds no padding bytes for alignment of common blocks.
records	Aligns derived-type components and record structure fields on default natural boundaries.
nosequence	Causes derived-type components declared with the SEQUENCE statement to be packed, regardless of current alignment rules set by the user.

By default, no padding is added to common blocks but padding is added to structures.

## Description

This option specifies the alignment to use for certain data items. The compiler adds padding bytes to perform the alignment.

Option	Description
align none	Tells the compiler not to add padding bytes anywhere in common
	blocks or structures. This is the same as specifying noalign.
align	Aligns all common block entities on natural boundaries up to 4
commons	bytes, by adding padding bytes as needed.
	The align nocommons option adds no padding to common
	blocks. In this case, unaligned data can occur unless the order of
	data items specified in the COMMON statement places the
	largest numeric data item first, followed by the next largest
	numeric data (and so on), followed by any character data.
align	Aligns all common block entities on natural boundaries up to 8

Option	Description
dcommons	bytes, by adding padding bytes as needed.  This option is useful for applications that use common blocks, unless your application has no unaligned data or, if the application might have unaligned data, all data items are four bytes or smaller. For applications that use common blocks where all data items are four bytes or smaller, you can specify /align:commons instead of /align:dcommons.  The align nodcommons option adds no padding to common blocks.  On Windows systems, if you specify the /stand:f90 or /stand:f95 option, /align:dcommons is ignored.  On Linux and Mac OS X systems, if you specify any -std option or the -stand f90 or -stand f95 option, -align dcommons is ignored.
align norecords	Aligns components of derived types and fields within record structures on arbitrary byte boundaries with no padding.  The align records option requests that multiple data items in record structures and derived-type structures without the SEQUENCE statement be naturally aligned, by adding padding as needed.
align rec <i>n</i> byte	Aligns components of derived types and fields within record structures on the smaller of the size boundary specified ( $n$ ) or the boundary that will naturally align them. $n$ can be 1, 2, 4, 8, or 16. When you specify this option, each structure member after the first is stored on either the size of the member type or $n$ -byte boundaries, whichever is smaller. For example, to specify 2 bytes as the packing boundary (or alignment constraint) for all structures and unions in the file prog1.f, use the following

Option	Description
	command:  ifort {-align rec2byte   /align:rec2byte} prog1.f  This option does not affect whether common blocks are naturally aligned or packed.
align	Aligns components of a derived type declared with the
sequence	SEQUENCE statement (sequenced components) according to
	the alignment rules that are currently in use. The default
	alignment rules are to align unsequenced components on natural
	boundaries.
	The align nosequence option requests that sequenced
	components be packed regardless of any other alignment rules.
	Note that align none implies align nosequence.
	If you specify an option for standards checking,
	/align:sequence is ignored.
align all	Tells the compiler to add padding bytes whenever possible to
	obtain the natural alignment of data items in common blocks,
	derived types, and record structures. Specifies align
	nocommons, align dcommons, align records, align
	nosequence. This is the same as specifying align with no
	keyword.

## **Alternate Options**

align	Linux and Mac OS X: -noalign
none	Windows: /noalign
align	Linux and Mac OS X: -align rec16byte, -Zp16
records	Windows: /align:rec16byte, /Zp16
align	Linux and Mac OS X: -Zp1, -align reclbyte
norecords	

Windows: /Zp1, /align:rec1byte

align Linux and Mac OS X:  $-zp\{1|2|4|8|16\}$ 

rec*n*byte Windows:  $/Zp{1|2|4|8|16}$ 

align all Linux and Mac OS X: -align commons -align

dcommons -align records -align nosequence

Windows:

/align:nocommons,dcommons,records,nosequence

#### See Also

Optimizing Applications: Setting Data Type and Alignment

## allow

Determines whether the compiler allows certain behaviors.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

#### **Syntax**

Linux and Mac OS X: -allow keyword

Windows: /allow:keyword

#### Arguments

keyword Specifies the behaviors to allow or disallow. Possible values are:

[no]fpp\_comments Determines how the fpp
preprocessor treats Fortran endof-line comments in preprocessor

directive lines.

#### Default

fpp\_comments The compiler recognizes Fortran-style end-of-line comments in preprocessor lines.

## Description

This option determines whether the compiler allows certain behaviors.

Option	Description
allow nofpp_comments	Tells the compiler to disallow Fortran-style end-of-
	line comments on preprocessor lines. Comment
	indicators have no special meaning.

### **Alternate Options**

None

#### Example

#### Consider the following:

```
#define MAX_ELEMENTS 100 ! Maximum number of elements
```

By default, the compiler recognizes Fortran-style end-of-line comments on preprocessor lines. Therefore, the line above defines MAX\_ELEMENTS to be "100" and the rest of the line is ignored. If allow nofpp\_comments is specified, Fortran comment conventions are not used and the comment indicator "!" has no special meaning. So, in the above example, "! Maximum number of elements" is interpreted as part of the value for the MAX\_ELEMENTS definition.

Option allow nofpp\_comments can be useful when you want to have a Fortran directive as a define value; for example:

```
#define dline(routname) !dec$ attributes alias:"__routname":: routname
```

## altparam

Allows alternate syntax (without parentheses) for PARAMETER statements.

## **IDE Equivalent**

Windows: Language > Enable Alternate PARAMETER Syntax

Linux: None

Mac OS X: Language > Enable Alternate PARAMETER Syntax

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -altparam

-noaltparam

Windows: /altparam

/noaltparam

## Arguments

None

#### Default

altparam The alternate syntax for PARAMETER statements is allowed.

## Description

This option specifies that the alternate syntax for PARAMETER statements is allowed. The alternate syntax is:

```
PARAMETER c = expr[, c = expr]...
```

This statement assigns a name to a constant (as does the standard PARAMETER statement), but there are no parentheses surrounding the assignment list.

In this alternative statement, the form of the constant, rather than implicit or explicit typing of the name, determines the data type of the variable.

## **Alternate Options**

```
altparam Linux and Mac OS X: -dps
```

Windows: /Qdps, /4Yaltparam

noaltparam Linux and Mac OS X: -nodps

Windows: /Qdps-, /4Naltparam

## ansi-alias, Qansi-alias

Tells the compiler to assume that the program adheres to Fortran Standard type aliasability rules.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -ansi-alias

-no-ansi-alias

Windows: Qansi-alias

Qansi-alias-

## Arguments

None

#### Default

```
-ansi- Programs adhere to Fortran Standard type aliasability rules.
```

alias

or

/Qansi-

alias

Description

This option tells the compiler to assume that the program adheres to type

aliasability rules defined in the Fortran Standard.

For example, an object of type real cannot be accessed as an integer. For

information on the rules for data types and data type constants, see "Data Types,"

Constants, and Variables" in the Language Reference.

This option directs the compiler to assume the following:

Arrays are not accessed out of arrays' bounds.

Pointers are not cast to non-pointer types and vice-versa.

References to objects of two different scalar types cannot alias. For example,

an object of type integer cannot alias with an object of type real or an object

of type real cannot alias with an object of type double precision.

If your program adheres to the Fortran Standard type aliasability rules, this option

enables the compiler to optimize more aggressively. If it doesn't adhere to these

rules, then you should disable the option with -no-ansi-alias (Linux and Mac

OS X) or /Qansi-alias- (Windows) so the compiler does not generate

incorrect code.

**Alternate Options** 

None

arch

Tells the compiler to generate optimized code specialized for the processor that

executes your program.

**IDE Equivalent** 

Windows: Code Generation > Enable Enhanced Instruction Set

Linux: None

Mac OS X: None

56

#### **Architectures**

IA-32, Intel® 64 architectures

## **Syntax**

Linux and Mac OS X: -archprocessor

Windows: /arch:processor

## **Arguments**

processor ls the processor type. Possible values are:

IA32 Generates code that will run on

any Pentium or later processor.

Disables any default extended

instruction settings, and any

previously set extended

instruction settings. This value

is only available on Linux and

Windows systems using IA-32

architecture.

SSE This is the same as specifying

IA32.

SSE2 Generates code for Intel®

Streaming SIMD Extensions 2

(Intel® SSE2). This value is

only available on Linux and

Windows systems.

SSE3 Generates code for Intel®

Streaming SIMD Extensions 3

(Intel® SSE3).

SSSE3 Generates code for Intel®

Supplemental Streaming SIMD

Extensions 3 (Intel® SSSE3).

SSE4.1 Generates code for Intel®

Streaming SIMD Extensions 4

Vectorizing Compiler and

Media Accelerators.

#### Default

Windows For more information on the default values, see Arguments

and Linux above.

systems:

SSE2

Mac OS X

systems

using IA-32

architecture:

SSE3

Mac OS X

systems

using Intel®

64

architecture:

SSSE3

## Description

This option tells the compiler to generate optimized code specialized for the processor that executes your program.

Code generated with the values IA32, SSE, SSE2 or SSE3 should execute on any compatible non-Intel processor with support for the corresponding instruction set.

For compatibility with Compaq\* Visual Fortran, the compiler allows the following keyword values. However, you should use the suggested replacements.

Compatibility Suggested Replacement

Value

pn1 -mia32 or /arch:IA32

pn2 -mia32 **or** /arch:IA32

pn3 -mia32 **or** /arch:IA32

pn4 -msse2 or /arch:SSE2

## **Alternate Options**

Linux and Mac OS X: -m

Windows: /architecture

#### See Also

 $\underline{x}$ ,  $\underline{Qx}$  compiler option  $\underline{ax}$ ,  $\underline{Qax}$  compiler option

m compiler option

## architecture

See arch.

## asmattr

Specifies the contents of an assembly listing file.

## **IDE Equivalent**

Windows: Output > Assembler Output

Linux: None

Mac OS X: None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: None

Windows: /asmattr:keyword

/noasmattr

### **Arguments**

keyword Specifies the contents of the assembly listing file. Possible

values are:

none Produces no assembly listing.

machine Produces an assembly listing with

machine code.

source Produces an assembly listing with

source code.

all Produces an assembly listing with

machine code and source code.

#### Default

/noasmattr No assembly listing is produced.

#### Description

This option specifies what information, in addition to the assembly code, should be generated in the assembly listing file.

To use this option, you must also specify option /asmfile, which causes an assembly listing to be generated.

Option	Description
/asmattr:none	Produces no assembly listing. This is the same as

Option	Description
	specifying /noasmattr.
/asmattr:machine	Produces an assembly listing with machine code.  The assembly listing file shows the hex machine instructions at the beginning of each line of assembly code.  The file cannot be assembled; the filename is the name of the source file with an extension of .cod.
/asmattr:source	Produces an assembly listing with source code.  The assembly listing file shows the source code as interspersed comments.  Note that if you use alternate option -fsource-asm, you must also specify the -S option.
/asmattr:all	Produces an assembly listing with machine code and source code.  The assembly listing file shows the source code as interspersed comments and shows the hex machine instructions at the beginning of each line of assembly code.  This file cannot be assembled.

## **Alternate Options**

/asmattr:none Linux and Mac OS X: None
Windows: /noasmattr

/asmattr:machine Linux and Mac OS X: -fcode-asm
Windows: /FAc

/asmattr:source Linux and Mac OS X: -fsource-asm
Windows: /FAs

/asmattr:all Linux and Mac OS X: None
Windows: /FAcs

## See Also

asmfile compiler option

## asmfile

Specifies that an assembly listing file should be generated.

## **IDE Equivalent**

Windows: Output > ASM Listing Name

Linux: None

Mac OS X: None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: None

Windows: /asmfile[:file | dir]

/noasmfile

## Arguments

file Is the name of the assembly listing file.

dir ls the directory where the file should be placed. It

can include file.

#### Default

/noasmfile No assembly listing file is produced.

## Description

This option specifies that an assembly listing file should be generated (optionally named file).

If *file* is not specified, the filename will be the name of the source file with an extension of *.asm*; the file is placed in the current directory.

### **Alternate Options**

Linux and Mac OS X: -S

Windows: /Fa

#### See Also

s compiler option

#### assume

Tells the compiler to make certain assumptions.

### **IDE Equivalent**

Windows: Compatibility > Treat Backslash as Normal Character in Strings

(/assume:[no]bscc)

**Data > Assume Dummy Arguments Share Memory Locations** 

(/assume:[no]dummy\_aliases)

**Data > Constant Actual Arguments Can Be Changed** 

(/assume:[no]protect\_constants)

Data > Use Bytes as RECL=Unit for Unformatted Files

(/assume:[no]byterecl)

Floating Point > Enable IEEE Minus Zero Support (/assume: [no]minus0)

Optimization > I/O Buffering (/assume: [no]buffered\_io)

Preprocessor > Default Include and Use Path

(/assume:[no]source\_include)

Preprocessor > OpenMP Conditional Compilation (/assume: [no]cc\_omp)

**External Procedures > Append Underscore to External Names** 

(/assume:[no]underscore)

Linux: None

Mac OS X: Optimization > I/O Buffering (-assume [no]buffered\_io)

Preprocessor > OpenMP Conditional Compilation (-assume [no]cc\_omp)

Preprocessor > Default Include and Use Path (-assume

[no]source\_include)

Compatibility > Treat Backslash as Normal Character in Strings (-assume [no]bscc)

Data > Assume Dummy Arguments Share Memory Locations (-assume
[no]dummy\_aliases)

Data > Constant Actual Arguments Can Be Changed (-assume

[no]protect\_constants)

Data > Use Bytes as RECL=Unit for Unformatted Files (-assume

[no]byterecl)

Floating Point > Enable IEEE Minus Zero Support (-assume [no]minus0)

External Procedures > Append Underscore to External Names (-assume

[no]underscore)

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -assume keyword

Windows: /assume:keyword

### Arguments

*keyword* Specifies the assumptions to be made. Possible values are:

none Disables all assume

options.

[no]bscc Determines whether the

backslash character is

treated as a C-style control

character syntax in character literals.

is immediately written to disk or accumulated in a

buffer.

[no]byterecl Determines whether units

for the OPEN statement RECL specifier (record

length) value in

unformatted files are in bytes or longwords (four-

byte units).

[no]cc\_omp Determines whether

conditional compilation as defined by the OpenMP Fortran API is enabled or

disabled.

[no]dummy\_aliases Determines whether the

compiler assumes that

dummy arguments to

procedures share memory

locations with other

dummy arguments or with COMMON variables that

are assigned.

[no]minus0 Determines whether the

compiler uses Fortran 95 or Fortran 90/77 standard

semantics in the SIGN

intrinsic when treating -0.0

and +0.0 as 0.0, and how

it writes the value on

formatted output.

[no]old\_boz Determines whether the

binary, octal, and

hexadecimal constant arguments in intrinsic functions INT, REAL, DBLE, and CMPLX are treated as signed integer

constants.

READs or WRITEs to

UNIT=\* go to stdin or

stdout, respectively.

[no]old\_xor Determines whether .XOR.

is defined by the compiler

as an intrinsic operator.

[no]protect\_constants Determines whether a

constant actual argument

or a copy of it is passed to

a called routine.

optimizer honors

parentheses in REAL and

**COMPLEX** expression

evaluations by not

reassociating operations.

allocatable objects on the

left-hand side of an

assignment are treated

according to Fortran 95/90

rules or Fortran 2003

rules.

[no]source\_include Determines whether the

compiler searches for USE

modules and INCLUDE

files in the default directory

or in the directory where

the source file is located.

[no]std\_mod\_proc\_name Determines whether the

names of module

procedures are allowed to conflict with user external

symbol names.

[no]underscore Determines whether the

compiler appends an

underscore character to

external user-defined

names.

[no]2underscores

(Linux and Mac OS X)

Determines whether the

compiler appends two

underscore characters to

external user-defined

names.

[no]writeable-strings Determines whether

character constants go into

non-read-only memory.

#### Default

nobscc The backslash character is treated as a

normal character in character literals.

nobuffered\_io Data in the internal buffer is immediately

written (flushed) to disk (OPEN specifier

BUFFERED='NO').

If you set the FORT\_BUFFERED environment

variable to true, the default is assume

buffered io.

nobyterecl Units for OPEN statement RECL values with

unformatted files are in four-byte (longword)

units.

nocc\_omp Conditional compilation as defined by the

OpenMP Fortran API is disabled unless option

-openmp (Linux) or /Qopenmp (Windows) is

specified.

If compiler option -openmp (Linux and Mac

OS X) or /Qopenmp (Windows) is specified,

the default is assume cc\_omp.

memory locations with other dummy

arguments or with variables shared through

use association, host association, or common

block use.

nominus 0 The compiler uses Fortran 90/77 standard

semantics in the SIGN intrinsic to treat -0.0

and +0.0 as 0.0, and writes a value of 0.0 with

no sign on formatted output.

noold\_boz The binary, octal, and hexadecimal constant

arguments in intrinsic functions INT, REAL, DBLE, and CMPLX are treated as bit strings that represent a value of the data type of the

intrinsic, that is, the bits are not converted.

old\_unit\_star The READs or WRITEs to UNIT=\* go to stdin

or stdout, respectively, even if UNIT=5 or 6

has been connected to another file.

old\_xor Intrinsic operator .XOR. is defined by the

compiler.

protect\_constants A constant actual argument is passed to a

called routine. Any attempt to modify it results

in an error.

noprotect\_parens The optimizer reorders REAL and COMPLEX

expressions without regard for parentheses if

it produces faster executing code.

norealloc\_lhs The compiler uses Fortran 95/90 rules when

interpreting assignment statements. The lefthand side is assumed to be allocated with the correct shape to hold the right-hand side. If it

is not, incorrect behavior will occur.

source\_include The compiler searches for USE modules and

INCLUDE files in the directory where the

source file is located.

nostd\_mod\_proc\_name The compiler allows the names of module procedures to conflict with user external symbol names. Windows: On Windows systems, the compiler does not append an underscore character to external nounderscore Linux and Mac OS X: user-defined names. On Linux and Mac OS X underscore systems, the compiler appends an underscore character to external user-defined names. no2underscores The compiler does not append two underscore (Linux and Mac OS X) characters to external user-defined names that contain an embedded underscore. nowriteable-strings The compiler puts character constants into read-only memory.

### Description

This option specifies assumptions to be made by the compiler.

Option	Description
assume none	Disables all the assume options.
assume bscc	Tells the compiler to treat the backslash character (\) as a C-style control (escape) character syntax in character literals. The "bscc" keyword means "BackSlashControlCharacters."
assume buffered_io	Tells the compiler to accumulate records in a buffer.  This sets the default for opening sequential output files to BUFFERED='YES', which also occurs if the FORT_BUFFERED run-time environment variable is specified.  When this option is specified, the internal buffer is

Option	Description
	filled, possibly by many record output statements (WRITE), before it is written to disk by the Fortran runtime system. If a file is opened for direct access, I/O buffering is ignored.  Using buffered writes usually makes disk I/O more efficient by writing larger blocks of data to the disk less often. However, if you request buffered writes, records not yet written to disk may be lost in the event of a system failure.  The OPEN statement BUFFERED specifier applies to a specific logical unit. In contrast, the assume [no]buffered_io option and the FORT_BUFFERED
assume byterecl	environment variable apply to all Fortran units.  Specifies that the units for the OPEN statement RECL specifier (record length) value are in bytes for unformatted data files, not longwords (four-byte units). For formatted files, the RECL value is always in bytes. If a file is open for unformatted data and assume byterecl is specified, INQUIRE returns RECL in bytes; otherwise, it returns RECL in longwords. An INQUIRE returns RECL in bytes if the unit is not open.
assume cc_omp	Enables conditional compilation as defined by the OpenMP Fortran API. That is, when "!\$space" appears in free-form source or "c\$spaces" appears in column 1 of fixed-form source, the rest of the line is accepted as a Fortran line.
assume dummy_aliases	Tells the compiler that dummy (formal) arguments to procedures share memory locations with other dummy

Option	Description
	arguments (aliases) or with variables shared through use association, host association, or common block use.
	Specify the option when you compile the called subprogram. The program semantics involved with dummy aliasing do not strictly obey the Fortran 95/90 standards and they slow performance, so you get better run-time performance if you do not use this option.
	However, if a program depends on dummy aliasing and you do not specify this option, the run-time behavior of the program will be unpredictable. In such programs, the results will depend on the exact optimizations that are performed. In some cases, normal results will occur, but in other cases, results will differ because the values used in computations involving the offending aliases will differ.
assume minus0	Tells the compiler to use Fortran 95 standard semantics for the treatment of the IEEE* floating value -0.0 in the SIGN intrinsic, which distinguishes the difference between -0.0 and +0.0, and to write a value of -0.0 with a negative sign on formatted output.
assume old_boz	Tells the compiler that the binary, octal, and hexadecimal constant arguments in intrinsic functions INT, REAL, DBLE, and CMPLX should be treated as signed integer constants.
assume noold_unit_star	Tells the compiler that READs or WRITEs to UNIT=* go to whatever file UNIT=5 or 6 is connected.

Option	Description
assume noold_xor	Prevents the compiler from defining .XOR. as an intrinsic operator. This lets you use .XOR. as a user-defined operator. This is a Fortran 2003 feature.
assume noprotect_constants	Tells the compiler to pass a copy of a constant actual argument. This copy can be modified by the called routine, even though the Fortran standard prohibits such modification. The calling routine does not see any modification to the constant.
assume protect_parens	Tells the optimizer to honor parentheses in REAL and COMPLEX expression evaluations by not reassociating operations. For example, (A+B)+C would not be evaluated as A+(B+C).  If assume noprotect_parens is specified, (A+B)+C would be treated the same as A+B+C and could be evaluated as A+(B+C) if it produced faster executing code.  Such reassociation could produce different results depending on the sizes and precision of the arguments.  For example, in (A+B)+C, if B and C had opposite signs and were very large in magnitude compared to A, A+B could result in the value as B; adding C would result in 0.0. With reassociation, B+C would be 0.0; adding A would result in a non-zero value.
assume realloc_lhs	Tells the compiler that when the left-hand side of an assignment is an allocatable object, it should be reallocated to the shape of the right-hand side of the assignment before the assignment occurs. This is the Fortran 2003 definition. This feature may cause extra

Option	Description
	overhead at run time.
assume nosource_include	Tells the compiler to search the default directory for module files specified by a USE statement or source files specified by an INCLUDE statement. This option affects fpp preprocessor behavior and the USE statement.
assume std_mod_proc_name	Tells the compiler to revise the names of module procedures so they do not conflict with user external symbol names. For example, procedure proc in module $m$ would be named $m_{MP_proc}$ . The Fortran 2003 Standard requires that module procedure names not conflict with other external symbols.  By default, procedure proc in module $m$ would be named $m_{mp_proc}$ , which could conflict with a user-defined external name $m_{mp_proc}$ .
assume underscore	Tells the compiler to append an underscore character to external user-defined names: the main program name, named common blocks, BLOCK DATA blocks, global data names in MODULEs, and names implicitly or explicitly declared EXTERNAL. The name of a blank (unnamed) common block remains _BLNK, and Fortran intrinsic names are not affected.
assume 2underscores (Linux and Mac OS X)	Tells the compiler to append two underscore characters to external user-defined names that contain an embedded underscore: the main program name, named common blocks, BLOCK DATA blocks, global data names in MODULEs, and names implicitly or explicitly declared EXTERNAL. The name of a blank

Option	Description
	(unnamed) common block remains _BLNK, and
	Fortran intrinsic names are not affected.
	This option does not affect external names that do not
	contain an embedded underscore. By default, the
	compiler only appends one underscore to those
	names. For example, if you specify assume
	2underscores for external names my_program and
	myprogram, my_program becomes my_program,
	but myprogram becomes myprogram
assume writeable-	Tells the compiler to put character constants into non-
strings	read-only memory.

### **Alternate Options**

assume nobscc Linux and Mac OS X: -nbs

Windows: /nbs

assume Linux and Mac OS X: -common-args

dummy\_aliases Windows:/Qcommon-args

assume Linux and Mac OS X: -us

underscore Windows: /us

assume Linux and Mac OS X: -nus

nounderscore Windows: None

# auto, Qauto

See automatic.

auto-scalar, Qauto-scalar

Causes scalar variables of intrinsic types INTEGER, REAL, COMPLEX, and LOGICAL that do not have the SAVE attribute to be allocated to the run-time stack.

### IDE Equivalent

Windows: Data > Local Variable Storage (/Qsave, /Qauto, /Qauto\_scalar)

Linux: None

Mac OS X: None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

### Syntax

Linux and Mac OS X: -auto-scalar

Windows: /Qauto-scalar

### Arguments

None

#### Default

```
-auto- Scalar variables of intrinsic types INTEGER, REAL,
```

scalar COMPLEX, and LOGICAL that do not have the SAVE attribute

or are allocated to the run-time stack. Note that if option

/Qauto-recursive, -openmp (Linux and Mac OS X), or /Qopenmp

scalar (Windows) is specified, the default is automatic.

### Description

This option causes allocation of scalar variables of intrinsic types INTEGER, REAL, COMPLEX, and LOGICAL to the run-time stack. It is as if they were declared with the AUTOMATIC attribute.

It does not affect variables that have the SAVE attribute (which include initialized locals) or that appear in an EQUIVALENCE statement or in a common block. This option may provide a performance gain for your program, but if your program depends on variables having the same value as the last time the routine was invoked, your program may not function properly. Variables that need to retain their values across subroutine calls should appear in a SAVE statement. You cannot specify option save, auto, or automatic with this option.



On Windows NT\* systems, there is a performance penalty for addressing a stack frame that is too large. This penalty may be incurred with <code>/automatic</code>, <code>/auto</code>, or <code>/Qauto</code> because arrays are allocated on the stack along with scalars. However, with <code>/Qauto-scalar</code>, you would have to have more than 32K bytes of local scalar variables before you incurred the performance penalty. <code>/Qauto-scalar</code> enables the compiler to make better choices about which variables should be kept in registers during program execution.

#### **Alternate Options**

None

#### See Also

auto compiler option
save compiler option

### autodouble, Qautodouble

See real-size.

#### automatic

Causes all local, non-SAVEd variables to be allocated to the run-time stack.

### **IDE Equivalent**

Windows: **Data > Local Variable Storage** 

Linux: None

Mac OS X: None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: -automatic

-noautomatic

Windows: /automatic

/noautomatic

### Arguments

None

#### Default

-auto- Scalar variables of intrinsic types INTEGER, REAL,

scalar COMPLEX, and LOGICAL are allocated to the run-time stack.

or Note that if one of the following options are specified, the

/Qauto- default is automatic:recursive, -openmp (Linux and Mac

scalar OS X), or /Qopenmp (Windows).

### Description

This option places local variables (scalars and arrays of all types), except those declared as SAVE, on the run-time stack. It is as if the variables were declared with the AUTOMATIC attribute.

It does not affect variables that have the SAVE attribute or ALLOCATABLE attribute, or variables that appear in an EQUIVALENCE statement or in a common block.

This option may provide a performance gain for your program, but if your program depends on variables having the same value as the last time the routine was invoked, your program may not function properly.

If you want to cause variables to be placed in static memory, specify option – save (Linux and Mac OS X) or /Qsave (Windows). If you want only scalar variables of certain intrinsic types to be placed on the run-time stack, specify option auto-scalar.



On Windows NT\* systems, there is a performance penalty for addressing a stack frame that is too large. This penalty may be incurred with <code>/automatic</code>, <code>/auto</code>, or <code>/Qauto</code> because arrays are allocated on the stack along with scalars. However, with <code>/Qauto-scalar</code>, you would have to have more than 32K bytes of local scalar variables before you incurred the performance penalty. <code>/Qauto-scalar</code> enables the compiler to make better choices about which variables should be kept in registers during program execution.

#### **Alternate Options**

automatic Linux and Mac OS X: -auto

Windows: /auto, /Qauto, /4Ya

noautomatic Linux and Mac OS X: -save, -noauto

Windows: /Qsave, /noauto, /4Na

#### See Also

auto-scalar compiler option
save, Qsave compiler option

ax, Qax

Tells the compiler to generate multiple, processor-specific auto-dispatch code paths for Intel processors if there is a performance benefit.

### **IDE Equivalent**

Windows: Code Generation > Add Processor-Optimized Code Path

Optimization > Generate Alternate Code Paths

Linux: None

Mac OS X: Code Generation > Add Processor-Optimized Code Path

#### Architectures

IA-32, Intel® 64 architectures

#### Syntax

Linux and Mac OS X: -axprocessor

Windows: /Qaxprocessor

#### Arguments

processor Indicates the processor for which code is

generated. The following descriptions refer to Intel® Streaming SIMD Extensions (Intel® SSE)

and Supplemental Streaming SIMD Extensions

(Intel® SSSE). Possible values are:

SSE4.2 Can generate Intel® SSE4

Efficient Accelerated String and

Text Processing instructions

supported by Intel® Core™ i7

processors. Can generate

Intel® SSE4 Vectorizing

Compiler and Media

Accelerator, Intel® SSSE3,

SSE3, SSE2, and SSE instructions and it can optimize for the Intel® Core™ processor family.

Vectorizing Compiler and
Media Accelerator instructions
for Intel processors. Can
generate Intel® SSSE3, SSE3,
SSE2, and SSE instructions
and it can optimize for Intel®
45nm Hi-k next generation
Intel® Core™
microarchitecture. This
replaces value S, which is
deprecated.

SSSE3 Can generate Intel® SSSE3,
SSE3, SSE2, and SSE
instructions for Intel processors
and it can optimize for the
Intel® Core™2 Duo processor
family. This replaces value T,
which is deprecated.

SSE3 Can generate Intel® SSE3,
SSE2, and SSE instructions for
Intel processors and it can
optimize for processors based
on Intel® Core™
microarchitecture and Intel

NetBurst® microarchitecture. This replaces value P, which is deprecated.

SSE2 Can generate Intel® SSE2 and

SSE instructions for Intel

processors, and it can optimize

for Intel® Pentium® 4

processors, Intel® Pentium® M processors, and Intel® Xeon® processors with Intel® SSE2. This value is not available on Mac OS X systems. This replaces value N, which is

deprecated.

#### Default

OFF No auto-dispatch code is generated. Processor-specific code is generated and is controlled by the setting of compiler option -m (Linux), compiler option /arch (Windows), or compiler option -x (Mac OS\* X).

#### Description

This option tells the compiler to generate multiple, processor-specific autodispatch code paths for Intel processors if there is a performance benefit. It also generates a baseline code path. The baseline code is usually slower than the specialized code.

The baseline code path is determined by the architecture specified by the -x(Linux and Mac OS X) or /Qx (Windows) option. While there are defaults for the -x or /0x option that depend on the operating system being used, you can specify an architecture for the baseline code that is higher or lower than the

default. The specified architecture becomes the effective minimum architecture for the baseline code path.

If you specify both the -ax and -x options (Linux and Mac OS X) or the /Qax and /Qx options (Windows), the baseline code will only execute on processors compatible with the processor type specified by the -x or /Qx option.

This option tells the compiler to find opportunities to generate separate versions of functions that take advantage of features of the specified Intel® processor. If the compiler finds such an opportunity, it first checks whether generating a processor-specific version of a function is likely to result in a performance gain. If this is the case, the compiler generates both a processor-specific version of a function and a baseline version of the function. At run time, one of the versions is chosen to execute, depending on the Intel processor in use. In this way, the program can benefit from performance gains on more advanced Intel processors, while still working properly on older processors.

You can use more than one of the processor values by combining them. For example, you can specify <code>-axSSE4.1,SSSE3</code> (Linux and Mac OS X) or <code>/QaxSSE4.1,SSSE3</code> (Windows). You cannot combine the old style, deprecated options and the new options. For example, you cannot specify <code>-axSSE4.1,T</code> (Linux and Mac OS X) or <code>/OaxSSE4.1,T</code> (Windows).

Previous values W and K are deprecated. The details on replacements are as follows:

- Mac OS X systems: On these systems, there is no exact replacement for W or K. You can upgrade to the default option -msse3 (IA-32 architecture) or option -mssse3 (Intel® 64 architecture).
- Windows and Linux systems: The replacement for W is -msse2 (Linux) or /arch: SSE2 (Windows). There is no exact replacement for K. However, on Windows systems, /Qaxk is interpreted as /arch: IA32; on Linux systems, -axk is interpreted as -mia32. You can also do one of the following:

Upgrade to option -msse2 (Linux) or option /arch: SSE2 (Windows). This
will produce one code path that is specialized for Intel® SSE2. It will not run

on earlier processors

Specify the two option combination -mia32 -axsse2 (Linux) or

/arch:IA32 /QaxSSE2 (Windows). This combination will produce an

executable that runs on any processor with IA-32 architecture but with an

additional specialized Intel® SSE2 code path.

The -ax and /Qax options enable additional optimizations not enabled with

option -m or option /arch.

**Alternate Options** 

None

See Also

x, Qx compiler option

m compiler option

arch compiler option

В

Specifies a directory that can be used to find include files, libraries, and

executables.

**IDE Equivalent** 

None

Architectures

IA-32, Intel® 64, IA-64 architectures

Syntax

Linux and Mac OS X: -Bdir

Windows: None

Arguments

84

dir

Is the directory to be used. If necessary, the compiler adds a directory separator character at the end of *dir*.

#### Default

OFF The compiler looks for files in the directories specified in your PATH environment variable.

### Description

This option specifies a directory that can be used to find include files, libraries, and executables.

The compiler uses *dir* as a prefix.

For include files, the dir is converted to -I/dir/include. This command is added to the front of the includes passed to the preprocessor.

For libraries, the dir is converted to -L/dir. This command is added to the front of the standard -L inclusions before system libraries are added.

For executables, if *dir* contains the name of a tool, such as 1d or as, the compiler will use it instead of those found in the default directories.

The compiler looks for include files in *dir* /include while library files are looked for in *dir*.

Another way to get the behavior of this option is to use the environment variable GCC\_EXEC\_PREFIX.

### **Alternate Options**

None

#### **Bdynamic**

Enables dynamic linking of libraries at run time.

#### **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

### Syntax

Linux: -Bdynamic

Mac OS X: None

Windows: None

### Arguments

None

#### Default

OFF Limited dynamic linking occurs.

### Description

This option enables dynamic linking of libraries at run time. Smaller executables are created than with static linking.

This option is placed in the linker command line corresponding to its location on the user command line. It controls the linking behavior of any library that is passed using the command line.

All libraries on the command line following option <code>-Bdynamic</code> are linked dynamically until the end of the command line or until a <code>-Bstatic</code> option is encountered. The <code>-Bstatic</code> option enables static linking of libraries.

### **Alternate Options**

None

### See Also

Bstatic compiler option

### bigobj

Increases the number of sections that an object file can contain.

#### **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: None

Windows: /bigobj

#### Arguments

None

### Default

OFF An object file can hold up to 65,536 (2\*\*16) addressable sections.

### Description

This option increases the number of sections that an object file can contain. It increases the address capacity to 4,294,967,296(2\*\*32).

An .obj file produced with this option can only be effectively passed to a linker that shipped in Microsoft Visual C++\* 2005 or later. Linkers shipped with earlier versions of the product cannot read .obj files of this size.

This option may be helpful for .obj files that can hold more sections, such as machine generated code.

### **Alternate Options**

None

#### bintext

Places a text string into the object file (.obj) being generated by the compiler.

### **IDE Equivalent**

Windows: Code Generation > Object Text String

Linux: None

Mac OS X: None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

### Syntax

Linux and Mac OS X: None

Windows: /bintext:string

/nobintext

### Arguments

string Is the text string to go into the object file.

#### Default

/nobintext No text string is placed in the object file.

#### Description

This option places a text string into the object file (.obj) being generated by the compiler. The string also gets propagated into the executable file.

For example, this option is useful if you want to place a version number or copyright information into the object and executable.

If the string contains a space or tab, the string must be enclosed by double quotation marks ("). A backslash (\) must precede any double quotation marks contained within the string.

If the command line contains multiple /bintext options, the last (rightmost) one is used.

### **Alternate Options**

Linux and Mac OS X: None Windows: /Vstring **Bstatic** Enables static linking of a user's library. **IDE Equivalent** None Architectures IA-32, Intel® 64, IA-64 architectures Syntax Linux: -Bstatic Mac OS X: None Windows: None Arguments None Default OFF Default static linking occurs. Description This option enables static linking of a user's library. This option is placed in the linker command line corresponding to its location on the user command line. It controls the linking behavior of any library that is passed using the command line. All libraries on the command line following option -Bstatic are linked statically until the end of the command line or until a -Bdynamic option is encountered.

The -Bdynamic option enables dynamic linking of libraries.

Alternate Options
None
See Also
Bdynamic compiler option
С
Prevents linking.
IDE Equivalent
None
Architectures
IA-32, Intel® 64, IA-64 architectures
Syntax
Linux and Mac OS X: -c
Windows: /c
Arguments
None
Default
OFF Linking is performed.
Description
This option prevents linking. Compilation stops after the object file is generated.
The compiler generates an object file for each Fortran source file.
Alternate Options
Linux and Mac OS X: None
Windows: /compile-only, /nolink

### C

See check.

#### CB

See check.

### ccdefault

Specifies the type of carriage control used when a file is displayed at a terminal screen.

### **IDE Equivalent**

Windows: Run-time > Default Output Carriage Control

Linux: None

Mac OS X: Run-time > Default Output Carriage Control

#### Architectures

IA-32, Intel® 64, IA-64 architectures

### Syntax

Linux and Mac OS X: -ccdefault keyword

Windows: /ccdefault:keyword

### Arguments

keyword Specifies the carriage-control setting to use. Possible values

are:

none Tells the compiler to use no carriage

control processing.

default Tells the compiler to use the default

carriage-control setting.

fortran Tells the compiler to use normal

Fortran interpretation of the first

character. For example, the character

0 causes output of a blank line before

a record.

list Tells the compiler to output one line

feed between records.

#### Default

ccdefault The compiler uses the default carriage control setting. default

#### Description

This option specifies the type of carriage control used when a file is displayed at a terminal screen (units 6 and \*). It provides the same functionality as using the CARRIAGECONTROL specifier in an OPEN statement.

The default carriage-control setting can be affected by the vms option. If vms is specified with ccdefault default, carriage control defaults to normal Fortran interpretation (ccdefault fortran) if the file is formatted and the unit is connected to a terminal. If novms (the default) is specified with ccdefault default, carriage control defaults to list (ccdefault list).

#### **Alternate Options**

None

#### check

Checks for certain conditions at run time.

#### **IDE Equivalent**

Windows:

Run-time > Runtime Error Checking (/nocheck, /check:all, or
/xcheck:none)

Run-time > Check Array and String Bounds (/check: [no]bounds)

Run-time > Check Uninitialized Variables (/check:[no]uninit)

Run-time > Check Edit Descriptor Data Type (/check: [no]format)

Run-time > Check Edit Descriptor Data Size

(/check:[no]output\_conversion)

Run-time > Check For Actual Arguments Using Temporary Storage

(/check:[no]arg\_temp\_created)

Run-time > Check For Null Pointers and Allocatable Array References

(/check:[no]pointers)

Linux: None

Mac OS X: Run-time > Runtime Error Checking (-check all, -check none)

Run-time > Check Array and String Bounds (-check [no]bounds)

Run-time > Check Edit Descriptor Data Type (-check [no]format)

Run-time > Check Edit Descriptor Data Size (-check

[no]output\_conversion)

Run-time > Check For Actual Arguments Using Temporary Storage (-check

[no]arg\_temp\_created)

Run-time > Check for Uninitialized Variables (-check [no]uninit)

Run-time > Check For Null Pointers and Allocatable Array References

(/check:[no]pointers)

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

#### **Syntax**

Linux and Mac OS X: -check [keyword]

-nocheck

Windows: /check[:keyword]

#### /nocheck

#### **Arguments**

*keyword* Specifies the conditions to check. Possible values are:

none Disables all check options.

[no]arg\_temp\_created Determines whether

checking occurs for actual arguments before routine

calls.

[no]bounds Determines whether

checking occurs for array subscript and character substring expressions.

[no]format Determines whether

checking occurs for the

data type of an item being

formatted for output.

[no]output\_conversion Determines whether

checking occurs for the fit

of data items within a

designated format

descriptor field.

[no]pointers Determines whether

checking occurs for certain

disassociated or

uninitialized pointers or unallocated allocatable

objects.

[no]uninit Determines whether

checking occurs for uninitialized variables.

all Enables all check options.

### Default

nocheck No checking is performed for run-time failures. Note that if option vms is specified, the defaults are check format and check output\_conversion.

### Description

This option checks for certain conditions at run time.

Option	Description
check none	Disables all check options (same as nocheck).
check	Enables run-time checking on whether actual arguments
arg_temp_created	are copied into temporary storage before routine calls. If a
	copy is made at run-time, an informative message is
	displayed.
check bounds	Enables compile-time and run-time checking for array
	subscript and character substring expressions. An error is
	reported if the expression is outside the dimension of the
	array or the length of the string.
	For array bounds, each individual dimension is checked.
	Array bounds checking is not performed for arrays that
	are dummy arguments in which the last dimension bound
	is specified as * or when both upper and lower dimensions
	are 1.
	Once the program is debugged, omit this option to reduce
	executable program size and slightly improve run-time

Option	Description
	performance.
check format	Issues the run-time FORVARMIS fatal error when the data type of an item being formatted for output does not match the format descriptor being used (for example, a REAL*4 item formatted with an I edit descriptor). With check noformat, the data item is formatted using the specified descriptor unless the length of the item cannot accommodate the descriptor (for example, it is still an error to pass an INTEGER*2 item to an E edit descriptor).
check output_conversion	Issues the run-time OUTCONERR continuable error message when a data item is too large to fit in a designated format descriptor field without loss of significant digits. Format truncation occurs, the field is filled with asterisks (*), and execution continues.
check pointers	Enables run-time checking for disassociated or uninitialized Fortran pointers, unallocated allocatable objects, and integer pointers that are uninitialized.
check uninit	Enables run-time checking for uninitialized variables. If a variable is read before it is written, a run-time error routine will be called. Only local scalar variables of intrinsic type INTEGER, REAL, COMPLEX, and LOGICAL without the SAVE attribute are checked.
check all	Enables all check options. This is the same as specifying check with no keyword.

To get more detailed location information about where an error occurred, use option traceback.

# **Alternate Options**

check none Linux and Mac OS X: -nocheck

Windows: /nocheck, /4Nb

check bounds Linux and Mac OS X: -CB

Windows: /CB

check uninit Linux and Mac OS X: -CU

Windows: /RTCu, /CU

check all Linux and Mac OS X: -check, -C

Windows: /check, /4Yb, /C

#### See Also

traceback compiler option

cm

See warn.

### common-args, Qcommon-args

See <u>assume</u>.

### compile-only

See c.

### complex-limited-range, Qcomplex-limited-range

Determines whether the use of basic algebraic expansions of some arithmetic operations involving data of type COMPLEX is enabled.

### IDE Equivalent

Windows: Floating point > Limit COMPLEX Range

Linux: None

Mac OS X: Floating point > Limit COMPLEX Range

### Architectures

IA-32, Intel® 64, IA-64 architectures

### Syntax

Linux and Mac OS X: -complex-limited-range

-no-complex-limited-range

Windows: /Qcomplex-limited-range

/Qcomplex-limited-range-

#### Arguments

None

#### Default

-no-complex-limited- Basic algebraic expansions of some

range arithmetic operations involving data of type

or/Qcomplex-limited- COMPLEX are disabled.

range-

### Description

This option determines whether the use of basic algebraic expansions of some arithmetic operations involving data of type COMPLEX is enabled.

When the option is enabled, this can cause performance improvements in programs that use a lot of COMPLEX arithmetic. However, values at the

extremes of the exponent range may not compute correctly.

### **Alternate Options**

None

### convert

Specifies the format of unformatted files containing numeric data.

### **IDE Equivalent**

Windows: Compatibility > Unformatted File Conversion

Linux: None

Mac OS X: Compatibility > Unformatted File Conversion

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: -convert keyword

Windows: /convert:keyword

### **Arguments**

keyword Specifies the format for the unformatted numeric data.

#### Possible values are:

native Specifies that unformatted data

should not be converted.

big\_endian Specifies that the format will be big

endian for integer data and big endian IEEE floating-point for real

and complex data.

cray Specifies that the format will be big

endian for integer data and CRAY\*

floating-point for real and complex

data.

Specifies that the format will be little
(Linux, Mac OS endian for integer data, and VAX
X) processor floating-point format
F\_floating, D\_floating, and X\_floating

for real and complex data.

fgx Specifies that the format will be little (Linux, Mac OS endian for integer data, and VAX X) processor floating-point format

F\_floating, G\_floating, and X\_floating

for real and complex data.

Specifies that the format will be big endian for integer data and IBM\*

System\370 floating-point format for

real and complex data.

little\_endian Specifies that the format will be little
endian for integer data and little
endian IEEE floating-point for real

and complex data.

vaxd Specifies that the format will be little

endian for integer data, and VAX\* processor floating-point format

F\_floating, D\_floating, and H\_floating

for real and complex data.

vaxg Specifies that the format will be little

endian for integer data, and VAX processor floating-point format

F\_floating, G\_floating, and H\_floating

for real and complex data.

Default

convert No conversion is performed on unformatted files containing native numeric data.

# Description

This option specifies the format of unformatted files containing numeric data.

Option	Description
convert native	Specifies that unformatted data should not be converted.
convert big_endian	Specifies that the format will be big endian for INTEGER*1, INTEGER*2, INTEGER*4, or INTEGER*8, and big endian IEEE floating-point for REAL*4, REAL*8, REAL*16, COMPLEX*8, COMPLEX*16, or COMPLEX*32.
convert cray	Specifies that the format will be big endian for INTEGER*1, INTEGER*2, INTEGER*4, or INTEGER*8, and CRAY* floating-point for REAL*8 or COMPLEX*16.
convert fdx	Specifies that the format will be little endian for INTEGER*1, INTEGER*2, INTEGER*4, or INTEGER*8, and VAX processor floating-point format F_floating for REAL*4 or COMPLEX*8, D_floating for REAL*8 or COMPLEX*16, and X_floating for REAL*16 or COMPLEX*32.
convert fgx	Specifies that the format will be little endian for INTEGER*1, INTEGER*2, INTEGER*4, or INTEGER*8, and VAX processor floating-point format F_floating for REAL*4 or COMPLEX*8, G_floating for REAL*8 or COMPLEX*16, and X_floating for REAL*16 or COMPLEX*32.
convert ibm	Specifies that the format will be big endian for INTEGER*1, INTEGER*2, or INTEGER*4, and IBM* System\370 floating-point format for REAL*4 or COMPLEX*8 (IBM short 4) and

Option	Description
	REAL*8 or COMPLEX*16 (IBM long 8).
convert little_endiar	Specifies that the format will be little endian for INTEGER*1, INTEGER*2, INTEGER*4, or INTEGER*8 and little endian IEEE floating-point for REAL*4, REAL*8, REAL*16, COMPLEX*8, COMPLEX*16, or COMPLEX*32.
convert vaxd	Specifies that the format will be little endian for INTEGER*1, INTEGER*2, INTEGER*4, or INTEGER*8, and VAX processor floating-point format F_floating for REAL*4 or COMPLEX*8, D_floating for REAL*8 or COMPLEX*16, and H_floating for REAL*16 or COMPLEX*32.
convert vaxg	Specifies that the format will be little endian for INTEGER*1, INTEGER*2, INTEGER*4, or INTEGER*8, and VAX processor floating-point format F_floating for REAL*4 or COMPLEX*8, G_floating for REAL*8 or COMPLEX*16, and H_floating for REAL*16 or COMPLEX*32.

# **Alternate Options**

None

срр, Осрр

See fpp, Qfpp.

CU

See check.

# cxxlib

Determines whether the compile links using the C++ run-time libraries provided by gcc.

IDE Equivalent
----------------

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -cxxlib[=dir]

-cxxlib-nostd

-no-cxxlib

Windows: None

# Arguments

dir Is an optional top-level location for the gcc binaries

and libraries.

# Default

-no-cxxlib The compiler uses the default run-time libraries

and does not link to any additional C++ run-time

libraries.

#### Description

This option determines whether the compile links using the C++ run-time libraries provided by gcc.

Option -cxxlib-nostd prevents the compiler from linking with the standard C++ library. It is only useful for mixed-language applications.

# **Alternate Options**

None

#### See Also

Building Applications: Options for Interoperability

D

Defines a symbol name that can be associated with an optional value.

# **IDE Equivalent**

Windows: **General > Preprocessor Definitions** 

**Preprocessor Definitions** 

**Preprocessor > Preprocessor Definitions to FPP only** 

Linux: None

Mac OS X: Preprocessor > Preprocessor Definitions

**Preprocessor > Preprocessor Definitions to FPP only** 

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -Dname[=value]

Windows: /Dname[=value]

#### Arguments

*name* Is the name of the symbol.

value Is an optional integer or an optional character

string delimited by double quotes; for example,

Dname=string.

### Default

noD Only default symbols or macros are defined.

# Description

Defines a symbol name that can be associated with an optional value.

This definition is used during preprocessing.

If a *value* is not specified, *name* is defined as "1".

If you want to specify more than one definition, you must use separate  $\ensuremath{\mathbb{D}}$  options.

If you specify noD, all preprocessor definitions apply only to fpp and not to Intel®

Fortran conditional compilation directives. To use this option, you must also

specify option fpp.



On Linux and Mac OS X systems, if you are not specifying a *value*, do not use D for *name*, because it will conflict with the -DD option.

### **Alternate Options**

D Linux and Mac OS X: None

Windows: /define:name[=value]

noD Linux and Mac OS X: -nodefine

Windows: /nodefine

### See Also

Building Applications: Predefined Preprocessor Symbols

# d-lines, Qd-lines

Compiles debug statements.

#### **IDE Equivalent**

Windows: Language > Compile Lines With D in Column 1

Linux: None

Mac OS X: Language > Compile Lines With D in Column 1

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -d-lines

-nod-lines

Windows: /d-lines

/nod-lines
/Qd-lines

# Arguments

None

#### Default

nod- Debug lines are treated as comment lines.

lines

# Description

This option compiles debug statements. It specifies that lines in fixed-format files that contain a D in column 1 (debug statements) should be treated as source code.

# **Alternate Options**

Linux and Mac OS X: -DD

Windows: None

# dbglibs

Tells the linker to search for unresolved references in a debug run-time library.

# **IDE Equivalent**

None

# Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: None

Windows: /dbglibs

/nodbglibs

# Arguments

None

# Default

/nodbglibs The linker does not search for unresolved references in a debug run-time library.

# Description

This option tells the linker to search for unresolved references in a debug runtime library.

The following table shows which options to specify for a debug run-time library:

Type of Library	Options Required	Alternate Option
Debug single-threaded	/libs:static /dbglibs	/MLd (this is a deprecated option)
Debug multithreaded	/libs:static /threads /dbglibs	/MTd
Multithreaded debug DLLs	/libs:dll /threads /dbglibs	/MDd
Debug Fortran QuickWin multi-thread applications	/libs:qwin /dbglibs	None

Type of Library Options Required Option
Option

Debug Fortran standard graphics (QuickWin /libs:qwins None

single-thread) applications /dbglibs

# **Alternate Options**

None

#### See Also

Building Applications: Specifying Consistent Library Types; Programming with Mixed Languages Overview

#### DD

See <u>dlines</u>, <u>Qdlines</u>.

# debug (Linux\* OS and Mac OS\* X)

Enables or disables generation of debugging information.

# **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

**Linux and Mac OS X**: -debug[keyword]

Windows: None

# Arguments

keyword Is the type of debugging information to be

generated. Possible values are:

none Disables generation of

debugging information.

full or all Generates complete

debugging information.

minimal Generates line number

information for

debugging.

[no]emit\_column Determines whether the

compiler generates

column number

information for debugging.

[no]inline- Determines whether the

debug-info compiler generates

enhanced debug

information for inlined

code.

[no]semantic-

stepping

Determines whether the

compiler generates

enhanced debug

information useful for

breakpoints and

stepping.

[no]variable-

Determines whether the

locations compiler generates

enhanced debug

information useful in

finding scalar local

variables.

extended Sets keyword values

semantic-stepping

and variablelocations.

For information on the non-default settings for these keywords, see the Description section.

# Default

-debug none No debugging information is generated.

# Description

This option enables or disables generation of debugging information.

Note that if you turn debugging on, optimization is turned off.

Keywords semantic-stepping, inline-debug-info, variable-locations, and extended can be used in combination with each other. If conflicting keywords are used in combination, the last one specified on the command line has precedence.

Option	Description
-debug none	Disables generation of debugging information.
-debug full <b>or</b> -	Generates complete debugging information. It is the
debug all	same as specifying -debug with no keyword.
-debug minimal	Generates line number information for debugging.
-debug	Generates column number information for debugging.
emit_column	
-debug inline-	Generates enhanced debug information for inlined code.
debug-info	It provides more information to debuggers for function
	call traceback.
-debug semantic-	Generates enhanced debug information useful for
stepping	breakpoints and stepping. It tells the debugger to stop
	only at machine instructions that achieve the final effect

Option	Description
	of a source statement.
	For example, in the case of an assignment statement,
	this might be a store instruction that assigns a value to a
	program variable; for a function call, it might be the
	machine instruction that executes the call. Other
	instructions generated for those source statements are
	not displayed during stepping.
	This option has no impact unless optimizations have also
	been enabled.
-debug variable-	Generates enhanced debug information useful in finding
locations	scalar local variables. It uses a feature of the Dwarf
	object module known as "location lists".
	This feature allows the run-time locations of local scalar
	variables to be specified more accurately; that is,
	whether, at a given position in the code, a variable value
	is found in memory or a machine register.
-debug extended	Sets keyword values semantic-stepping and
	variable-locations. It also tells the compiler to
	include column numbers in the line information.

On Linux\* systems, debuggers read debug information from executable images. As a result, information is written to object files and then added to the executable by the linker. On Mac OS\* X systems, debuggers read debug information from object files. As a result, the executables don't contain any debug information. Therefore, if you want to be able to debug on these systems, you must retain the object files.

# **Alternate Options**

For debug full, - Linux and Mac OS X: -g

debug all, or - Windows: /debug:full, /debug:all, or

debug /debug

For -debug Linux and Mac OS X: -inline-debug-info

inline-debug- (this is a deprecated option)

info Windows: None

#### See Also

debug (Windows\*) compiler option

**Building Applications: Debugging Overview** 

# debug (Windows\* OS)

Enables or disables generation of debugging information.

# IDE Equivalent

Windows: **General > Debug Information Format** (/debug:minimal, /debug:full)

Linux: None

Mac OS X: None

# Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: None

Windows: /debug[:keyword]

/nodebug

# Arguments

keyword Is the type of debugging information to be

generated. Possible values are:

none Generates no symbol table

information.

full or all Generates complete debugging information.

minimal Generates line numbers and minimal debugging information.

partial Deprecated Generates global symbol table information

needed for linking.

For information on the non-default settings for these keywords, see the Description section.

#### Default

/debug:none	This is the default on the command line and for a
	release configuration in the IDE.
/debug:full	This is the default for a debug configuration in the
	IDE.

# Description

This option enables or disables generation of debugging information. It is passed to the linker.

Note that if you turn debugging on, optimization is turned off.

If conflicting keywords are used in combination, the last one specified on the command line has precedence.

Option	Description
/debug:none	Disables generation of debugging
	information. It is the same as specifying
	/nodebug.
/debug:full or /debug:all	Generates complete debugging

Option	Description
	information. It produces symbol table information needed for full symbolic debugging of unoptimized code and global symbol information needed for linking. It is the same as specifying /debug with no keyword. If you specify /debug:full for an application that makes calls to C library routines and you need to debug calls into the C library, you should also specify /dbglibs to request that the appropriate C debug library be linked
/debug:minimal	against.  Generates line number information for debugging. It produces global symbol information needed for linking, but not local symbol table information needed for debugging.
/debug:partial	Generates global symbol table information needed for linking, but not local symbol table information needed for debugging. This option is deprecated and is not available in the IDE.

# **Alternate Options**

For Linux and Mac OS X: None

/debug:minimal Windows: /Zd (this is a deprecated option)

For /debug:full Linux and Mac OS X: None

Windows: /Zi, /Z7

or

/debug

# See Also

dbglibs compiler option

debug (Linux\* and Mac OS\* X) compiler option

Building Applications: Debugging Fortran Programs

# debug-parameters

Tells the compiler to generate debug information for PARAMETERs used in a program.

# IDE Equivalent

Windows: **Debugging > Information for PARAMETER Constants** 

Linux: None

Mac OS X: **Debug > Information for PARAMETER Constants** 

# Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -debug-parameters [keyword]

-nodebug-parameters

Windows: /debug-parameters[:keyword]

/nodebug-parameters

# Arguments

keyword Are the PARAMETERs to generate debug information for.

Possible values are:

none Generates no debug information for

any PARAMETERs used in the program. This is the same as specifying nodebug-parameters.

used Generates of

Generates debug information for only PARAMETERs that have actually been referenced in the program. This is the

default if you do not specify a

keyword.

all Generates debug information for all

PARAMETERs defined in the program.

#### Default

nodebug- The compiler generates no debug information for any parameters PARAMETERs used in the program. This is the same as specifying *keyword* none.

#### Description

This option tells the compiler to generate debug information for PARAMETERs used in a program.

Note that if a .mod file contains PARAMETERs, debug information is only generated for the PARAMETERs that have actually been referenced in the program, even if you specify *keyword* all.

# **Alternate Options**

None

# define

See D.

# diag, Qdiag

Controls the display of diagnostic information.

# IDE Equivalent

Windows: **Diagnostics > Disable Specific Diagnostics** (/Qdiag-disable id)

Diagnostics > Level of Static Analysis (/Qdiag-enable[:sv1,sv2, sv3])

Linux: None

Mac OS X: Diagnostics > Disable Specific Diagnostics (-diag-disable id)

Diagnostics > Level of Static Analysis (-diag-enable [sv1,sv2, sv3])

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -diag-type diag-list

Windows: /Qdiag-type:diag-list

# Arguments

type Is an action to perform on diagnostics. Possible

values are:

enable Enables a diagnostic message

or a group of messages.

disable Disables a diagnostic message

or a group of messages.

error Tells the compiler to change

diagnostics to errors.

warning Tells the compiler to change

diagnostics to warnings.

remark Tells the compiler to change

diagnostics to remarks

(comments).

diag-list Is a diagnostic group or ID value. Possible values

are:

driver Specifies diagnostic

messages issued by the

compiler driver.

vec Specifies diagnostic

messages issued by the

vectorizer.

par Specifies diagnostic

messages issued by the

auto-parallelizer (parallel

optimizer).

sv[n] Specifies diagnostic

messages issued by the Static Verifier. *n* can be any of the following: 1, 2, 3. For more details on

these values, see below.

warn Specifies diagnostic

messages that have a "warning" severity level.

error Specifies diagnostic

messages that have an

"error" severity level.

remark Specifies diagnostic

messages that are remarks or comments.

cpu-dispatch Specifies the CPU
dispatch remarks for
diagnostic messages.
These remarks are
enabled by default. This
diagnostic group is only
available on IA-32
architecture and Intel® 64
architecture.

id[,id,...] Specifies the ID number of one or more messages. If you specify more than one message number, they must be separated by commas. There can be no intervening white space between each id.

hag[,tag,...] Specifies the mnemonic
name of one or more
messages. If you specify
more than one mnemonic
name, they must be
separated by commas.
There can be no
intervening white space
between each tag.

Default

OFF The compiler issues certain diagnostic messages by default.

### Description

This option controls the display of diagnostic information. Diagnostic messages are output to stderr unless compiler option <code>-diag-file</code> (Linux and Mac OS X) or <code>/Qdiag-file</code> (Windows) is specified.

When *diag-list* value "warn" is used with the Static Verifier (sv) diagnostics, the following behavior occurs:

- Option -diag-enable warn (Linux and Mac OS X) and /Qdiagenable:warn (Windows) enable all Static Verifier diagnostics except those that have an "error" severity level. They enable all Static Verifier warnings, cautions, and remarks.
- Option -diag-disable warn (Linux and Mac OS X) and /Qdiag-disable:warn (Windows) disable all Static Verifier diagnostics except those that have an "error" severity level. They suppress all Static Verifier warnings, cautions, and remarks.

The following table shows more information on values you can specify for diaglist item sv.

diag-list Item	Descrip	tion
sv[n]	The valu	e of $n$ for Static Verifier messages can be any of the following:
	1	Produces the diagnostics with severity level set to all critical errors.
	2	Produces the diagnostics with severity level set to all errors. This is the default if n is not specified.
	3	Produces the diagnostics with severity level set to all errors and warnings.

To control the diagnostic information reported by the vectorizer, use the -vec-report (Linux and Mac OS X) or /Qvec-report (Windows) option.

To control the diagnostic information reported by the auto-parallelizer, use the – par–report (Linux and Mac OS X) or /Qpar–report (Windows) option.

### **Alternate Options**

enable vec Linux and Mac OS X: -vec-report

Windows: /Qvec-report

disable vec Linux and Mac OS X: -vec-report0

Windows: /Qvec-report0

enable par Linux and Mac OS X: -par-report

Windows: /Qpar-report

disable par Linux and Mac OS X: -par-report0

Windows: /Qpar-report0

#### **Example**

The following example shows how to enable diagnostic IDs 117, 230 and 450:

```
-diag-enable 117,230,450 ! Linux and Mac OS X systems /Qdiag-enable:117,230,450 ! Windows systems
```

The following example shows how to change vectorizer diagnostic messages to warnings:

```
-diag-enable vec -diag-warning vec ! Linux and Mac OS X systems /Qdiag-enable:vec /Qdiag-warning:vec ! Windows systems
```

Note that you need to enable the vectorizer diagnostics before you can change them to warnings.

The following example shows how to disable all auto-parallelizer diagnostic messages:

```
-diag-disable par ! Linux and Mac OS X systems /Qdiag-disable:par ! Windows systems
```

The following example shows how to produce Static Verifier diagnostic messages for all critical errors:

```
-diag-enable svl ! Linux and Mac OS X systems /Qdiag-enable:svl ! Windows system
```

The following example shows how to cause Static Verifier diagnostics (and default diagnostics) to be sent to a file:

```
-diag-enable sv -diag-file=stat_ver_msg ! Linux and Mac OS X systems /Qdiag-enable:sv /Qdiag-file:stat_ver_msg ! Windows systems
```

Note that you need to enable the Static Verifier diagnostics before you can send them to a file. In this case, the diagnostics are sent to file stat\_ver\_msg.diag. If a

file name is not specified, the diagnostics are sent to name-of-the-first-source-file.diag.

The following example shows how to change all diagnostic warnings and remarks to errors:

```
-diag-error warn, remark ! Linux and Mac OS X systems /Qdiag-error:warn, remark ! Windows systems
```

#### See Also

```
diag-dump, Qdiag-dump compiler option
diag-id-numbers, Qdiag-id-numbers compiler option
diag-file, Qdiag-file compiler option
par-report, Qpar-report compiler option
vec-report, Qvec-report compiler option
```

# diag-dump, Qdiag-dump

Tells the compiler to print all enabled diagnostic messages and stop compilation.

### **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -diag-dump

Windows: /Qdiag-dump

### Arguments

None

#### Default

OFF The compiler issues certain diagnostic messages by default.

# Description

This option tells the compiler to print all enabled diagnostic messages and stop compilation. The diagnostic messages are output to stdout.

This option prints the enabled diagnostics from all possible diagnostics that the compiler can issue, including any default diagnostics.

If -diag-enable *diag-list* (Linux and Mac OS X) or /Qdiag-enable *diag-list* (Windows) is specified, the print out will include the *diag-list* diagnostics.

# **Alternate Options**

None

#### Example

The following example adds vectorizer diagnostic messages to the printout of default diagnostics:

```
-diag-enable vec -diag-dump ! Linux and Mac OS X systems /Qdiag-enable:vec /Qdiag-dump ! Windows systems
```

#### See Also

diag, Qdiag compiler option

# diag-enable sv-include, Qdiag-enable sv-include

Tells the Static Verifier to analyze include files and source files when issuing diagnostic messages.

# **IDE Equivalent**

Windows: Diagnostics > Analyze Include Files

Linux: None

Mac OS X: Diagnostics > Analyze Include Files

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

#### **Syntax**

Linux and Mac OS X: -diag-enable sv-include

Windows: /Qdiag-enable sv-include

#### Arguments

None

#### **Default**

OFF The compiler issues certain diagnostic messages by default. If the Static Verifier is enabled, include files are not analyzed by default.

# Description

This option tells the Static Verifier to analyze include files and source files when issuing diagnostic messages. Normally, when Static Verifier diagnostics are enabled, only source files are analyzed.

To use this option, you must also specify <code>-diag-enable sv</code> (Linux and Mac OS X) or <code>/Qdiag-enable:sv</code> (Windows) to enable the Static Verifier diagnostics.

#### **Alternate Options**

None

#### Example

The following example shows how to cause include files to be analyzed as well as source files:

```
-diag-enable sv -diag-enable sv-include ! Linux and Mac OS systems /Qdiag-enable:sv /Qdiag-enable:sv-include ! Windows systems In the above example, the first compiler option enables Static Verifier messages. The second compiler option causes include files referred to by the source file to
```

The second compiler option causes include files referred to by the source file to be analyzed also.

### See Also

diag, Qdiag compiler option

# diag-error-limit, Qdiag-error-limit

Specifies the maximum number of errors allowed before compilation stops.

# IDE Equivalent

Windows: Compilation Diagnostics > Error Limit

Linux: None

Mac OS X: Compiler Diagnostics > Error Limit

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -diag-error-limitn

-no-diag-error-limit

Windows: /Odiag-error-limit:n

/Qdiag-error-limit-

#### Arguments

*n* Is the maximum number of error-level or fatal-level

compiler errors allowed.

### Default

A maximum of 30 error-level and fatal-level

messages are allowed.

#### Description

This option specifies the maximum number of errors allowed before compilation stops. It indicates the maximum number of error-level or fatal-level compiler errors allowed for a file specified on the command line.

If you specify -no-diag-error-limit (Linux and Mac OS X) or /Qdiag-error-limit- (Windows) on the command line, there is no limit on the number of errors that are allowed.

If the maximum number of errors is reached, a warning message is issued and the next file (if any) on the command line is compiled.

# Alternate Options

Linux and Mac OS X: -error-limit and -noerror-limit

Windows: /error-limit and /noerror-limit

# diag-file, Qdiag-file

Causes the results of diagnostic analysis to be output to a file.

# **IDE Equivalent**

Windows: **Diagnostics > Diagnostics File** 

Linux: None

Mac OS X: Diagnostics > Diagnostics File

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -diag-file[=file]

Windows: /Qdiag-file[:file]

# Arguments

file Is the name of the file for output.

# Default

OFF Diagnostic messages are output to stderr.

#### Description

This option causes the results of diagnostic analysis to be output to a file. The file is placed in the current working directory.

If *file* is specified, the name of the file is *file.diag*. The file can include a file extension; for example, if *file.ext* is specified, the name of the file is *file.ext*. If *file* is not specified, the name of the file is *name-of-the-first-source-file.diag*. This is also the name of the file if the name specified for file conflicts with a source file name provided in the command line.



If you specify <code>-diag-file</code> (Linux and Mac OS X) or <code>/Qdiag-file</code> (Windows) and you also specify <code>-diag-file-append</code> (Linux and Mac OS X) or <code>/Qdiag-file-append</code> (Windows), the last option specified on the command line takes precedence.

### **Alternate Options**

None

#### Example

The following example shows how to cause diagnostic analysis to be output to a file named my\_diagnostics.diag:

```
-diag-file=my_diagnostics ! Linux and Mac OS X systems /Qdiag-file:my_diagnostics ! Windows systems
```

# See Also

diag, Qdiag compiler option
diag-file-append, Qdiag-file-append compiler option

# diag-file-append, Qdiag-file-append

Causes the results of diagnostic analysis to be appended to a file.

#### **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -diag-file-append[=file]

Windows: /Qdiag-file-append[:file]

# Arguments

file Is the name of the file to be appended to. It can include a path.

#### Default

OFF Diagnostic messages are output to stderr.

# Description

This option causes the results of diagnostic analysis to be appended to a file. If you do not specify a path, the driver will look for *file* in the current working directory.

If *file* is not found, then a new file with that name is created in the current working directory. If the name specified for file conflicts with a source file name provided in the command line. the name of the file is *name-of-the-first-source-file.diag*.



If you specify <code>-diag-file-append</code> (Linux and Mac OS X) or <code>/Qdiag-file-append</code> (Windows) and you also specify <code>-diag-file</code> (Linux and Mac OS X) or <code>/Qdiag-file</code> (Windows), the last option specified on the command line takes precedence.

#### **Alternate Options**

None

#### Example

The following example shows how to cause diagnostic analysis to be appended to a file named my\_diagnostics.txt:

```
-diag-file-append=my_diagnostics.txt ! Linux and Mac OS X systems /Qdiag-file-append:my diagnostics.txt ! Windows systems
```

#### See Also

```
diag, Qdiag compiler option
diag-file, Qdiag-file compiler option
```

# diag-id-numbers, Qdiag-id-numbers

Determines whether the compiler displays diagnostic messages by using their ID number values.

# **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

#### **Syntax**

Linux and Mac OS X: -diag-id-numbers

-no-diag-id-numbers

Windows: /Qdiag-id-numbers

/Qdiag-id-numbers-

#### Arguments

None

#### Default

-diag-id- The compiler displays diagnostic messages by

numbers using their ID number values.

```
or/Qdiag-id-
numbers
```

### Description

This option determines whether the compiler displays diagnostic messages by using their ID number values. If you specify -no-diag-id-numbers (Linux and Mac OS X) or /Qdiag-id-numbers- (Windows), mnemonic names are output for driver diagnostics only.

# **Alternate Options**

None

# See Also

diag, Qdiag compiler option

# diag-once, Qdiag-once

Tells the compiler to issue one or more diagnostic messages only once.

# **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

```
Linux and Mac OS X: -diag-onceid[,id,...]

Windows: /Qdiag-once:id[,id,...]
```

# Arguments

id Is the ID number of the diagnostic message. If you specify more than one message number, they must be separated by commas.

There can be no intervening white space between each id.
Default
OFF The compiler issues certain diagnostic messages by default.
Description
This option tells the compiler to issue one or more diagnostic messages only once.
Alternate Options
None
dll
Specifies that a program should be linked as a dynamic-link (DLL) library.
IDE Equivalent
None
Architectures
IA-32, Intel® 64, IA-64 architectures
Syntax
Linux and Mac OS X: None
Windows: /dll
Arguments
None
Default

The program is not linked as a dynamic-link (DLL) library.

# Description

OFF

This option specifies that a program should be linked as a dynamic-link (DLL) library instead of an executable (.exe) file. It overrides any previous specification of run-time routines to be used and enables the /libs:dll option.

If you use this option with the /libs:qwin or /libs:qwins option, the compiler issues a warning.

# **Alternate Options**

Linux and Mac OS X: None

Windows: /LD

# double-size

Specifies the default KIND for DOUBLE PRECISION and DOUBLE COMPLEX variables.

# **IDE Equivalent**

Windows: Data > Default Double Precision KIND

Linux: None

Mac OS X: Data > Default Double Precision KIND

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -double-size size

Windows: /double-size:size

# Arguments

Size Specifies the default KIND for DOUBLE PRECISION and DOUBLE COMPLEX declarations, constants, functions, and intrinsics. Possible values are: 64 (KIND=8) or 128 (KIND=16).

# Default

DOUBLE PRECISION variables are defined as REAL\*8 and DOUBLE COMPLEX variables are defined as COMPLEX\*16.

# Description

This option defines the default KIND for DOUBLE PRECISION and DOUBLE COMPLEX declarations, constants, functions, and intrinsics.

Option	Description
double-size	Defines DOUBLE PRECISION declarations, constants, functions,
64	and intrinsics as REAL(KIND=8) (REAL*8) and defines DOUBLE
	COMPLEX declarations, functions, and intrinsics as
	COMPLEX(KIND=8) (COMPLEX*16).
double-size	Defines DOUBLE PRECISION declarations, constants, functions,
128	and intrinsics as REAL(KIND=16) (REAL*16) and defines
	DOUBLE COMPLEX declarations, functions, and intrinsics as
	COMPLEX(KIND=16) (COMPLEX*32).

# **Alternate Options**

None

# dps, Qdps

See altparam.

# dryrun

Specifies that driver tool commands should be shown but not executed.

# **IDE Equivalent**

None

### Architectures

Intel® Fortran Compiler User and Reference Guides IA-32, Intel® 64, IA-64 architectures **Syntax** Linux and Mac OS X: -dryrun Windows: None Arguments None Default OFF No tool commands are shown, but they are executed. Description This option specifies that driver tool commands should be shown but not executed. **Alternate Options** None See Also v compiler option dumpmachine Displays the target machine and operating system configuration. **IDE Equivalent** None **Architectures** 

Syntax

IA-32, Intel® 64, IA-64 architectures

Linux a	and Mac OS X: -dumpmachine
Windo	ws: None
Argume	ents
None	
NOHE	
Default	
OFF	The compiler does not display target machine or operating system information.
Descrip	otion
_	otion displays the target machine and operating system configuration. No ation is performed.
Alterna	te Options
None	
dynam	nic-linker
Specifi	ies a dynamic linker other than the default.
IDE Equ	uivalent
None	
Archite	ctures
IA-32,	Intel® 64, IA-64 architectures
Syntax	
Linux:	-dynamic-linker file

Arguments

Mac OS X:

Windows:

None

None

file	Is the name of the dynamic linker to be used.
Default	
OFF The default dy	namic linker is used.
Description	
This option lets you s	pecify a dynamic linker other than the default.
Alternate Options	
None	
dynamiclib	
	command to generate dynamic libraries.
invokes the TIDCOOT	command to generate dynamic libraries.
IDE Equivalent	
None	
Architectures	
IA-32, Intel® 64 archi	tectures
Syntax	
Linux:	None
Mac OS X:	-dynamiclib
Windows:	None
Arguments	
None	
Default	
OFF The compiler	produces an executable.
Description	

This option invokes the libtool command to generate dynamic libraries.

When passed this option, the compiler uses the libtool command to produce a dynamic library instead of an executable when linking.

To build static libraries, you should specify option -staticlib or libtool - static <objects>.

### **Alternate Options**

None

#### See Also

staticlib compiler option

# dyncom, Qdyncom

Enables dynamic allocation of common blocks at run time.

# **IDE Equivalent**

Windows: **Data > Dynamic Common Blocks** 

Linux: None

Mac OS X: Data > Dynamic Common Blocks

#### Architectures

IA-32, Intel® 64, IA-64 architectures

#### **Syntax**

```
Linux and Mac OS X: -dyncom "common1, common2, ..."

Windows: /Qdyncom "common1, common2, ..."
```

#### Arguments

common1,common2,... Are the names of the common blocks to be dynamically allocated. The list of names must be within quotes.

#### Default

OFF Common blocks are not dynamically allocated at run time.

# Description

This option enables dynamic allocation of the specified common blocks at run time. For example, to enable dynamic allocation of common blocks a, b, and c at run time, use this syntax:

```
/Qdyncom "a,b,c" ! on Windows systems
-dyncom "a,b,c" ! on Linux and Mac OS X systems
```

The following are some limitations that you should be aware of when using this option:

- An entity in a dynamic common cannot be initialized in a DATA statement.
- Only named common blocks can be designated as dynamic COMMON.
- An entity in a dynamic common block must not be used in an EQUIVALENCE expression with an entity in a static common block or a DATA-initialized variable.

# **Alternate Options**

None

#### See Also

Building Applications: Allocating Common Blocks

Ε

Causes the preprocessor to send output to stdout.

**IDE Equivalent** 

None

**Architectures** 

IA-32, Intel® 64, IA-64 architectures

**Syntax** 

Linux and Mac OS X: -E
Windows: /E
Arguments
None
Default
OFF Preprocessed source files are output to the compiler.
Description
This option causes the preprocessor to send output to stdout. Compilation
stops when the files have been preprocessed.
When you specify this option, the compiler's preprocessor expands your source module and writes the result to stdout. The preprocessed source contains #line
directives, which the compiler uses to determine the source file and line number.
and all of the complication of the control and
Alternate Options
None
None
None e90, e95, e03
None  e90, e95, e03  See <u>warn</u> .
None  e90, e95, e03  See <u>warn</u> .  EP
None  e90, e95, e03  See warn.  EP  Causes the preprocessor to send output to stdout, omitting #line directives.
e90, e95, e03 See warn.  EP Causes the preprocessor to send output to stdout, omitting #line directives.  IDE Equivalent
e90, e95, e03 See warn.  EP Causes the preprocessor to send output to stdout, omitting #line directives.  IDE Equivalent None

Intel® Fortran Compiler User and Reference Guides Linux and Mac OS X: -EP Windows: /EP Arguments None Default OFF Preprocessed source files are output to the compiler. Description This option causes the preprocessor to send output to stdout, omitting #line directives. If you also specify option preprocess-only, option P, or option F, the preprocessor will write the results (without #line directives) to a file instead of stdout. **Alternate Options** None error-limit See diag-error-limit, Qdiag-error-limit. exe Specifies the name for a built program or dynamic-link library. IDE Equivalent

None

Architectures

IA-32, Intel® 64, IA-64 architectures

Linux and Mac OS X: None

Windows: /exe:{file | dir}

Arguments

file Is the name for the built program or dynamic-link

library.

dir Is the directory where the built program or

dynamic-link library should be placed. It can

include file.

Default

OFF The name of the file is the name of the first source file on the

command line with file extension .exe, so file.f becomes file.exe.

Description

This option specifies the name for a built program (.EXE) or a dynamic-link library

(.DLL).

You can use this option to specify an alternate name for an executable file. This

is especially useful when compiling and linking a set of input files. You can use

the option to give the resulting file a name other than that of the first input file

(source or object) on the command line.

You can use this option to specify an alternate name for an executable file. This

is especially useful when compiling and linking a set of input files. You can use

the option to give the resulting file a name other than that of the first input file

(source or object) on the command line.

**Alternate Options** 

Linux and Mac OS X: -o

Windows: /Fe

Example

141

The following example creates a dynamic-link library file named file.dll (note that you can use /LD in place of /dll):

```
ifort /dll /exe:file.dll a.f
```

In the following example (which uses the alternate option /Fe), the command produces an executable file named outfile.exe as a result of compiling and linking three files: one object file and two Fortran source files.

```
prompt>ifort /Feoutfile.exe file1.obj file2.for file3.for

By default, this command produces an executable file named file1.exe.
```

#### See Also

o compiler option

#### extend-source

Specifies the length of the statement field in a fixed-form source file.

## **IDE Equivalent**

Windows: Language > Fixed Form Line Length

Linux: None

Mac OS X: Language > Fixed Form Line Length

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -extend-source [size]

-noextend-source

Windows: /extend-source[:size]

/noextend-source

### Arguments

size Is the length of the statement field in a fixed-form source file.

Possible values are: 72, 80, or 132.

## Default

72	If you do not specify this option or you specify
	noextend-source, the statement field ends at
	column 72.
132	If you specify extend_source without size, the
	statement field ends at column 132

## Description

This option specifies the size (column number) of the statement field of a source line in a fixed-form source file. This option is valid only for fixed-form files; it is ignored for free-form files.

When size is specified, it is the last column parsed as part of the statement field. Any columns after that are treated as comments.

If you do not specify <code>size</code>, it is the same as specifying <code>extend\_source 132</code>.

Option	Description
extend-	Specifies that the statement field ends at column 72.
source 72	
extend-	Specifies that the statement field ends at column 80.
source 80	
extend-	Specifies that the statement field ends at column 132.
source 132	

## **Alternate Options**

extend-source	Linux and Mac OS X: -72
72	Windows: /4L72
extend-source	Linux and Mac OS X: -80
80	Windows: /4L80
extend-source	Linux and Mac OS X: -132

Windows: /Qextend-source, /4L132

#### extfor

Specifies file extensions to be processed by the compiler as Fortran files.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: None

Windows: /extfor:ext

## Arguments

ext Are the file extensions to be processed as a Fortran file.

#### Default

OFF Only the file extensions recognized by the compiler are processed as Fortran files. For more information, see *Building Applications*.

#### Description

This option specifies file extensions (ext) to be processed by the compiler as Fortran files. It is useful if your source file has a nonstandard extension. You can specify one or more file extensions. A leading period before each extension is optional; for example, /extfor:myf95 and /extfor:myf95 are equivalent.

### **Alternate Options**

None

#### See Also

source compiler option Tells the compiler to compile the file as a Fortran source file.

## extfpp

Specifies file extensions to be recognized as a file to be preprocessed by the Fortran preprocessor.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: None

Windows: /extfpp:ext

### Arguments

ext Are the file extensions to be preprocessed by the Fortran preprocessor.

#### Default

OFF Only the file extensions recognized by the compiler are preprocessed by fpp. For more information, see *Building Applications*.

## Description

This option specifies file extensions (ext) to be recognized as a file to be preprocessed by the Fortran preprocessor (fpp). It is useful if your source file has a nonstandard extension.

You can specify one or more file extensions. A leading period before each extension is optional; for example, /extfpp:myfpp and /extfpp:.myfpp are equivalent.

## **Alternate Options**

None

### extlnk

Specifies file extensions to be passed directly to the linker.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

#### **Syntax**

Linux and Mac OS X: None

Windows: /extlnk:ext

## Arguments

*ext* Are the file extensions to be passed directly to the linker.

## Default

OFF Only the file extensions recognized by the compiler are passed to the linker. For more information, see Building Applications.

## Description

This option specifies file extensions (*ext*) to be passed directly to the linker. It is useful if your source file has a nonstandard extension.

You can specify one or more file extensions. A leading period before each extension is optional; for example, /extlnk:myobj and /extlnk:.myobj are equivalent.

## **Alternate Options**

None

## F (Windows\*)

Specifies the stack reserve amount for the program.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: None

Windows: /Fn

## Arguments

*n* Is the stack reserve amount. It can be specified as

a decimal integer or by using a C-style convention

for constants (for example, /F0x1000).

#### Default

OFF The stack size default is chosen by the operating system.

## Description

This option specifies the stack reserve amount for the program. The amount (n) is passed to the linker.

Note that the linker property pages have their own option to do this.

## **Alternate Options**

None

## f66

Tells the compiler to apply FORTRAN 66 semantics.

## **IDE Equivalent**

Windows: Language > Enable FORTRAN 66 Semantics

Linux: None

Mac OS X: Language > Enable FORTRAN 66 Semantics

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -f66

Windows: /f66

## Arguments

None

### Default

OFF The compiler applies Fortran 95 semantics.

## Description

This option tells the compiler to apply FORTRAN 66 semantics when interpreting language features. This causes the following to occur:

- DO loops are always executed at least once
- FORTRAN 66 EXTERNAL statement syntax and semantics are allowed

- If the OPEN statement STATUS specifier is omitted, the default changes to STATUS='NEW' instead of STATUS='UNKNOWN'
- If the OPEN statement BLANK specifier is omitted, the default changes to BLANK='ZERO' instead of BLANK='NULL'

## **Alternate Options**

Linux and Mac OS X: -66

Windows: None

#### f77rtl

Tells the compiler to use the run-time behavior of FORTRAN 77.

## **IDE Equivalent**

Windows: Compatibility > Enable F77 Run-Time Compatibility

Linux: None

Mac OS X: Compatibility > Enable F77 Run-Time Compatibility

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -f77rtl

-nof77rtl

Windows: /f77rtl

/nof77rtl

## Arguments

None

#### Default

nof77rtl The compiler uses the run-time behavior of Intel® Fortran.

## Description

This option tells the compiler to use the run-time behavior of FORTRAN 77. Specifying this option controls the following run-time behavior:

- When the unit is not connected to a file, some INQUIRE specifiers will return different values:
- NUMBER= returns 0
- ACCESS= returns 'UNKNOWN'
- BLANK= returns 'UNKNOWN'
- FORM= returns 'UNKNOWN'
- PAD= defaults to 'NO' for formatted input.
- NAMELIST and list-directed input of character strings must be delimited by apostrophes or quotes.
- When processing NAMELIST input:
- Column 1 of each record is skipped.
- The '\$' or '&' that appears prior to the group-name must appear in column 2 of the input record.

## **Alternate Options**

None

## Fa

See asmfile

#### FA

See <u>asmattr</u>

## falias

Determines whether aliasing should be assumed in the program.

## **IDE Equivalent**

None

## Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -falias

-fno-alias

Windows: None

## Arguments

None

#### Default

-falias Aliasing is assumed in the program.

## Description

This option determines whether aliasing should be assumed in the program.

If you do not want aliasing to be assumed in the program, specify -fno-alias.

## **Alternate Options**

Linux and Mac OS X: None

Windows: /Oa[-]

#### See Also

ffnalias compiler option

## falign-functions, Qfnalign

Tells the compiler to align functions on an optimal byte boundary.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64 architectures

## Syntax

Linux and Mac OS X: -falign-functions[=n]

-fno-align-functions

Windows: /Qfnalign[:n]

/Qfnalign-

## Arguments

*n* Is the byte boundary for function alignment.

Possible values are 2 or 16.

#### Default

-fno- The compiler aligns functions on 2-byte boundaries. This

align- is the same as specifying -falign-functions=2

functions (Linux and Mac OS X) or /Qfnalign: 2 (Windows).

or

/Qfnalign-

## Description

This option tells the compiler to align functions on an optimal byte boundary. If you do not specify n, the compiler aligns the start of functions on 16-byte boundaries.

#### **Alternate Options**

None

## falign-stack

Tells the compiler the stack alignment to use on entry to routines.

## **IDE Equivalent**

None

#### **Architectures**

IA-32 architecture

## **Syntax**

Linux and Mac OS X: -falign-stack=mode

Windows: None

## Arguments

mode Is the method to use for stack alignment. Possible

values are:

default Tells the compiler to use

default heuristics for stack alignment. If alignment is required, the compiler

dynamically aligns the stack.

maintain- Tells the compiler to not

16-byte assume any specific stack

alignment, but attempt to

maintain alignment in case the

stack is already aligned. If alignment is required, the

compiler dynamically aligns the

stack. This setting is compatible with GCC.

assume- Tells the compiler to assume

16-byte the stack is aligned on 16-byte

boundaries and continue to maintain 16-byte alignment.

This setting is compatible with GCC.

#### Default

-falign- The compiler uses default heuristics for stack stack=default alignment.

## Description

This option tells the compiler the stack alignment to use on entry to routines.

## **Alternate Options**

None

## fast

Maximizes speed across the entire program.

## **IDE Equivalent**

Windows: None

Linux: None

Mac OS X: None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -fast

Windows: /fast

## Arguments

None

#### Default

OFF The optimizations that maximize speed are not enabled.

## Description

This option maximizes speed across the entire program.

It sets the following options:

On systems using IA-64 architecture:

```
Windows: /03 and /Qipo
Linux: -ipo, -03, and -static
```

• On systems using IA-32 architecture and Intel® 64 architecture:

```
Mac OS X: -ipo, -mdynamic-no-pic, -03, -no-prec-div, -static, and -xHost
```

```
Windows: /03, /Qipo, /Qprec-div-, and /QxHost
Linux: -ipo, -03, -no-prec-div, -static, and -xHost
```

When option fast is specified on systems using IA-32 architecture or Intel® 64 architecture, you can override the -xHost or /QxHost setting by specifying a different processor-specific -x or /Qx option on the command line. However, the last option specified on the command line takes precedence.

For example, if you specify -fast -xSSE3 (Linux) or /fast /QxSSE3 (Windows), option -xSSE3 or /QxSSE3 takes effect. However, if you specify -xSSE3 -fast (Linux) or /QxSSE3 /fast (Windows), option -xHost or /OxHost takes effect.



The options set by option fast may change from release to release.

#### **Alternate Options**

None

#### fast-transcendentals, Qfast-transcendentals

Enables the compiler to replace calls to transcendental functions with faster but less precise implementations.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

#### **Syntax**

```
Linux and Mac OS X: -fast-transcendentals
```

-no-fast-transcendentals

Windows: /Qfast-transcendentals

/Qfast-transcendentals-

#### Default

```
The default depends on the setting of -fp-model
transcendentals (Linux and Mac OS X) or /fp (Windows).

Or /Qfast- The default is ON if default setting -fp-model
transcendentals fast or /fp:fast is in effect. However, if a value-
safe option such as -fp-model precise or
/fp:precise is specified, the default is OFF.
```

#### Description

This option enables the compiler to replace calls to transcendental functions with implementations that may be faster but less precise.

It tells the compiler to perform certain optimizations on transcendental functions, such as replacing individual calls to sine in a loop with a single call to a less precise vectorized sine library routine.

This option has an effect only when specified with one of the following options:

• Windows\* OS: /fp:except or /fp:precise

• Linux\* OS and Mac OS\* X: -fp-model except or -fp-model precise
You cannot use this option with option -fp-model strict (Linux and Mac OS
X) or /fp:strict (Windows).

## **Alternate Options**

None

#### See Also

fp-model, fp compiler option

#### fcode-asm

Produces an assembly listing with machine code annotations.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -fcode-asm

Windows: None

## Arguments

None

#### Default

OFF No machine code annotations appear in the assembly listing file, if one is produced.

## Description

This option produces an assembly listing file with machine code annotations.

The assembly listing file shows the hex machine instructions at the beginning of each line of assembly code. The file cannot be assembled; the filename is the name of the source file with an extension of .cod.

To use this option, you must also specify option -S, which causes an assembly listing to be generated.

## **Alternate Options**

Linux and Mac OS X: None

Windows: /asmattr:machine

#### See Also

S compiler option

Fe

See exe

## fexceptions

Enables exception handling table generation.

## IDE Equivalent

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -fexceptions

-fno-exceptions

Windows: None

## **Arguments**

None

Default

-fno-

Exception handling table generation is disabled.

exceptions

Description

This option enables C++ exception handling table generation, preventing Fortran routines in mixed-language applications from interfering with exception handling between C++ routines. The -fno-exceptions option disables C++ exception handling table generation, resulting in smaller code. When this option is used, any use of C++ exception handling constructs (such as try blocks and throw

statements) when a Fortran routine is in the call chain will produce an error.

**Alternate Options** 

None

ffnalias

Specifies that aliasing should be assumed within functions.

**IDE Equivalent** 

None

**Architectures** 

IA-32, Intel® 64, IA-64 architectures

**Syntax** 

Linux and Mac OS X: -ffnalias

-fno-fnalias

Windows: None

Arguments

None

159

## Default

-ffnalias Aliasing is assumed within functions.

## Description

This option specifies that aliasing should be assumed within functions.

The -fno-fnalias option specifies that aliasing should not be assumed within functions, but should be assumed across calls.

## **Alternate Options**

Linux and Mac OS X: None

Windows: /Ow[-]

#### See Also

falias compiler option

FI

See fixed.

## finline

Tells the compiler to inline functions declared with cDEC\$ ATTRIBUTES FORCEINLINE.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -finline

-fno-inline

Arguments			
None			
Default			
-fno- inline	The compiler does not inline functions declared with cDEC\$ ATTRIBUTES FORCEINLINE.		
Description			
This option tells the compiler to inline functions declared with cDEC\$ ATTRIBUTES FORCEINLINE.			
Alternate Optio	ns		
None			
finline-functions			
Enables function inlining for single file compilation.			
IDE Equivalent			
None			
Architectures			
IA-32, Intel® 64, IA-64 architectures			
Syntax			
Linux and Mac OS X: -finline-functions			
	-fno-inline-functions		
Windows:	None		
Arguments			
None			

Windows:

None

### Default

```
-finline- Interprocedural optimizations occur. However, if you functions specify -00, the default is OFF.
```

## Description

This option enables function inlining for single file compilation.

It enables the compiler to perform inline function expansion for calls to functions defined within the current source file.

The compiler applies a heuristic to perform the function expansion. To specify the size of the function to be expanded, use the <code>-finline-limit</code> option.

## **Alternate Options**

```
Linux and Mac OS X: -inline-level=2
```

Windows: /Ob2

#### See Also

```
ip, Qip compiler option
finline-limit compiler option
```

#### finline-limit

Lets you specify the maximum size of a function to be inlined.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -finline-limit=n

Windows: None

#### **Arguments**

n

Must be an integer greater than or equal to zero. It is the maximum number of lines the function can have to be considered for inlining.

#### Default

OFF The compiler uses default heuristics when inlining functions.

## Description

This option lets you specify the maximum size of a function to be inlined. The compiler inlines smaller functions, but this option lets you inline large functions. For example, to indicate a large function, you could specify 100 or 1000 for n. Note that parts of functions cannot be inlined, only whole functions. This option is a modification of the -finline-functions option, whose behavior occurs by default.

## **Alternate Options**

None

#### See Also

finline-functions compiler option

## finstrument-functions, Qinstrument-functions

Determines whether routine entry and exit points are instrumented.

#### **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -finstrument-functions

-fno-instrument-functions

Windows: /Qinstrument-functions

/Qinstrument-functions-

#### Arguments

None

#### Default

```
-fno- Routine entry and exit points are not instrumented.
instrument-
functions
or/Qinstrument-
functions-
```

### Description

This option determines whether routine entry and exit points are instrumented. It may increase execution time.

The following profiling functions are called with the address of the current routine and the address of where the routine was called (its "call site"):

- This function is called upon routine entry:
- On IA-32 architecture and Intel® 64 architecture:

```
void __cyg_profile_func_enter (void *this_fn,
void *call_site);
```

On IA-64 architecture:

```
void __cyg_profile_func_enter (void **this_fn,
void *call_site);
```

- This function is called upon routine exit:
- On IA-32 architecture and Intel® 64 architecture:

```
void __cyg_profile_func_exit (void *this_fn,
void *call_site);
```

On IA-64 architecture:

```
void __cyg_profile_func_exit (void **this_fn,
void *call_site);
```

On IA-64 architecture, the additional de-reference of the function pointer argument is required to obtain the routine entry point contained in the first word of the routine descriptor for indirect routine calls. The descriptor is documented in the Intel® Itanium® Software Conventions and Runtime Architecture Guide, section 8.4.2. You can find this design guide at web site http://www.intel.com These functions can be used to gather more information, such as profiling information or timing information. Note that it is the user's responsibility to provide these profiling functions.

If you specify -finstrument-functions (Linux and Mac OS X) or /Qinstrument-functions (Windows), routine inlining is disabled. If you specify -fno-instrument-functions or /Qinstrument-functions-, inlining is not disabled.

This option is provided for compatibility with gcc.

#### **Alternate Options**

None

#### fixed

Specifies source files are in fixed format.

## **IDE Equivalent**

Windows: Language > Source File Format (/free, /fixed)

Linux: None

Mac OS X: Language > Source File Format (/free, /fixed)

#### Architectures

IA-32, Intel® 64, IA-64 architectures

#### **Syntax**

Linux and Mac OS X: -fixed

-nofixed

Windows: /fixed

/nofixed

#### Arguments

None

#### Default

OFF The source file format is determined from the file extension.

## Description

This option specifies source files are in fixed format. If this option is not specified, format is determined as follows:

- Files with an extension of .f90, .F90, or .i90 are free-format source files.
- Files with an extension of .f, .for, .FOR, .ftn, .FTN, .fpp, .FPP, or .i are fixed-format files.

Note that on Linux and Mac OS X systems, file names and file extensions are case sensitive.

## **Alternate Options**

Linux and Mac OS X: -FI

Windows: /nofree, /FI, /4Nf

## fkeep-static-consts, Qkeep-static-consts

Tells the compiler to preserve allocation of variables that are not referenced in the source.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -fkeep-static-consts

-fno-keep-static-consts

Windows: /Qkeep-static-consts

/Qkeep-static-consts-

### **Arguments**

None

#### Default

-fno-keep- If a variable is never referenced in a routine, the

static-consts or variable is discarded unless optimizations are

/Qkeep-static- disabled by option -00 (Linux and Mac OS X) or

consts- /Od (Windows).

#### Description

This option tells the compiler to preserve allocation of variables that are not referenced in the source.

The negated form can be useful when optimizations are enabled to reduce the memory usage of static data.

## **Alternate Options**

None

## fltconsistency

Enables improved floating-point consistency.

## **IDE Equivalent**

Windows: Floating-Point > Floating-Point Consistency

Linux: None

Mac OS X: Floating-Point > Improve Floating-Point Consistency

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

#### **Syntax**

Linux and Mac OS X: -fltconsistency

-nofltconsistency

Windows: /fltconsistency

/nofltconsistency

#### Arguments

None

#### Default

nofltconsistency Improved floating-point consistency is not enabled.

This setting provides better accuracy and run-time performance at the expense of less consistent floating-point results.

## Description

This option enables improved floating-point consistency and may slightly reduce execution speed. It limits floating-point optimizations and maintains declared precision. It also disables inlining of math library functions.

Floating-point operations are not reordered and the result of each floating-point operation is stored in the target variable rather than being kept in the floating-point processor for use in a subsequent calculation.

For example, the compiler can change floating-point division computations into multiplication by the reciprocal of the denominator. This change can alter the results of floating-point division computations slightly.

Floating-point intermediate results are kept in full 80 bits internal precision.

Additionally, all spills/reloads of the X87 floating point registers are done using

the internal formats; this prevents accidental loss of precision due to spill/reload behavior over which you have no control.

Specifying this option has the following effects on program compilation:

- On systems using IA-32 architecture or Intel® 64 architecture, floating-point user variables are not assigned to registers.
- On systems using IA-64 architecture, floating-point user variables may be assigned to registers. The expressions are evaluated using precision of source operands. The compiler will not use the Floating-point Multiply and Add (FMA) function to contract multiply and add/subtract operations in a single operation. The contractions can be enabled by using -IPF\_FMA (Linux) or /QIPF\_fma (Windows) option. The compiler will not speculate on floating-point operations that may affect the floating-point state of the machine.
- Floating-point arithmetic comparisons conform to IEEE 754.
- The exact operations specified in the code are performed. For example, division is never changed to multiplication by the reciprocal.
- The compiler performs floating-point operations in the order specified without reassociation.
- The compiler does not perform constant folding on floating-point values.
   Constant folding also eliminates any multiplication by 1, division by 1, and addition or subtraction of 0. For example, code that adds 0.0 to a number is executed exactly as written. Compile-time floating-point arithmetic is not performed to ensure that floating-point exceptions are also maintained.
- Whenever an expression is spilled, it is spilled as 80 bits (extended precision), not 64 bits (DOUBLE PRECISION). When assignments to type REAL and DOUBLE PRECISION are made, the precision is rounded from 80 bits down to 32 bits (REAL) or 64 bits (DOUBLE PRECISION). When you do not specify /op, the extra bits of precision are not always rounded away before the variable is reused.
- Even if vectorization is enabled by the -x (Linux and Mac OS X) or /Qx (Windows) options, the compiler does not vectorize reduction loops (loops computing the dot product) and loops with mixed precision types. Similarly,

the compiler does not enable certain loop transformations. For example, the compiler does not transform reduction loops to perform partial summation or loop interchange.

This option causes performance degradation relative to using default floatingpoint optimization flags.

On Windows systems, an alternative is to use the /Qprec option, which should provide better than default floating-point precision while still delivering good floating-point performance.

The recommended method to control the semantics of floating-point calculations is to use option -fp-model (Linux and Mac OS X) or /fp (Windows).

## **Alternate Options**

fltconsistency Linux and Mac OS X: -mp (this is a deprecated

option), -mieee-fp

Windows: /Op (this is a deprecated option)

nofltconsistency Linux and Mac OS X: -mno-ieee-fp

Windows: None

#### See Also

mp1, Qprec compiler option

fp-model, fp compiler option

Building Applications: Using Compiler Optimizations

#### Fm

This option has been deprecated.

See map.

## fma, Qfma

Enables the combining of floating-point multiplies and add/subtract operations.

## **IDE Equivalent**

Windows: Floating Point > Contract Floating-Point Operations

Linux: None

Mac OS X: None

#### **Architectures**

IA-64 architecture

## **Syntax**

Linux: -fma

-no-fma

Mac OS X: None

Windows: /Qfma

/Qfma-

## Arguments

None

#### Default

-fma Floating-point multiplies and add/subtract operations are

or/Qfma combined.

However, if you specify -mp (Linux), /Op (Windows),

/fp:strict (Windows), or -fp-model strict (Linux) but do not explicitly specify -fma or /Qfma, the default is

-no-fma **or** /Qfma-.

## Description

This option enables the combining of floating-point multiplies and add/subtract operations.

It also enables the contraction of floating-point multiply and add/subtract operations into a single operation. The compiler contracts these operations whenever possible.

## **Alternate Options**

Linux: -IPF-fma (this is a deprecated option)

Windows: /QIPF-fma (this is a deprecated option)

## See Also

fp-model, fp compiler option

Floating-point Operations: Floating-point Options Quick Reference

## fmath-errno

Tells the compiler that errno can be reliably tested after calls to standard math library functions.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -fmath-errno

-fno-math-errno

Windows: None

## Arguments

None

#### Default

 $\hbox{-fno-math-errno} \quad \hbox{The compiler assumes that the program does not} \\$ 

test errno after calls to standard math library

functions.

## Description

This option tells the compiler to assume that the program tests errno after calls

to math library functions. This restricts optimization because it causes the

compiler to treat most math functions as having side effects.

Option -fno-math-errno tells the compiler to assume that the program does

not test errno after calls to math library functions. This frequently allows the

compiler to generate faster code. Floating-point code that relies on IEEE

exceptions instead of errno to detect errors can safely use this option to

improve performance.

**Alternate Options** 

None

fminshared

Specifies that a compilation unit is a component of a main program and should

not be linked as part of a shareable object.

**IDE Equivalent** 

None

Architectures

IA-32, Intel® 64, IA-64 architectures

**Syntax** 

Linux and Mac OS X: -fminshared

Windows:

None

Arguments

None

Default

OFF Source files are compiled together to form a single object file.

173

Description

This option specifies that a compilation unit is a component of a main program

and should not be linked as part of a shareable object.

This option allows the compiler to optimize references to defined symbols without

special visibility settings. To ensure that external and common symbol references

are optimized, you need to specify visibility hidden or protected by using the -

fvisibility, -fvisibility-hidden, or -fvisibility-protected

option.

Also, the compiler does not need to generate position-independent code for the

main program. It can use absolute addressing, which may reduce the size of the

global offset table (GOT) and may reduce memory traffic.

**Alternate Options** 

None

See Also

fvisibility compiler option

fnsplit, Qfnsplit

Enables function splitting.

**IDE Equivalent** 

None

Architectures

/Qfnsplit[-]: IA-32 architecture, Intel® 64 architecture

-[no-]fnsplit: IA-64 architecture

Syntax

Linux: -fnsplit

-no-fnsplit

174

Mac OS X: None

Windows: /Qfnsplit

/Qfnsplit-

### Arguments

None

#### Default

-no-fnsplit Function splitting is not enabled unless -prof-

or/Qfnsplit- use (Linux) or /Qprof-use (Windows) is also

specified.

## Description

This option enables function splitting if <code>-prof-use</code> (Linux) or <code>/Qprof-use</code> (Windows) is also specified. Otherwise, this option has no effect. It is enabled automatically if you specify <code>-prof-use</code> or <code>/Qprof-use</code>. If you do not specify one of those options, the default is <code>-no-fnsplit</code> (Linux) or <code>/Qfnsplit-</code> (Windows), which disables function splitting but leaves function grouping enabled.

To disable function splitting when you use -prof-use or /Qprof-use, specify -no-fnsplit or /Qfnsplit-.

## **Alternate Options**

None

### See Also

Optimizing Applications:

Basic PGO Options

Example of Profile-Guided Optimization

## fomit-frame-pointer, Oy

Determines whether EBP is used as a general-purpose register in optimizations.

## **IDE Equivalent**

Windows: **Optimization > Omit Frame Pointers** 

Linux: None

Mac OS X: Optimization > Provide Frame Pointer

#### Architectures

```
-f[no-]omit-frame-pointer: IA-32 architecture, Intel® 64 architecture /Oy[-]: IA-32 architecture
```

## **Syntax**

Linux and Mac OS X: -fomit-frame-pointer

-fno-omit-frame-pointer

Windows: /Oy

/Oy-

## Arguments

None

#### Default

-fomit-frame- EBP is used as a general-purpose register in

pointer optimizations. However, on Linux\* and Mac OS X

or /Oy systems, the default is -fno-omit-frame-

pointer if option -00 or -g is specified. On

Windows\* systems, the default is /Oy- if option

/od is specified.

### Description

These options determine whether EBP is used as a general-purpose register in

optimizations. Options -fomit-frame-pointer and /Oy allow this use.

Options - fno-omit-frame-pointer and /Oy- disallow it.

Some debuggers expect EBP to be used as a stack frame pointer, and cannot

produce a stack backtrace unless this is so. The -fno-omit-frame-pointer

and /Oy- options direct the compiler to generate code that maintains and uses

EBP as a stack frame pointer for all functions so that a debugger can still

produce a stack backtrace without doing the following:

• For -fno-omit-frame-pointer: turning off optimizations with -00

For /Oy-: turning off /O1, /O2, or /O3 optimizations

The -fno-omit-frame-pointer option is set when you specify option -00 or

the -q option. The -fomit-frame-pointer option is set when you specify

option -01, -02, or -03.

The /Oy option is set when you specify the /O1, /O2, or /O3 option. Option

/Oy- is set when you specify the /Od option.

Using the -fno-omit-frame-pointer or /Oy- option reduces the number of

available general-purpose registers by 1, and can result in slightly less efficient

code.

**Alternate Options** 

Linux and Mac OS X: -fp (this is a deprecated option)

Windows: None

Fo

See object

fomit-frame-pointer, Oy

Determines whether EBP is used as a general-purpose register in optimizations.

**IDE Equivalent** 

Windows: Optimization > Omit Frame Pointers

177

Linux: None

Mac OS X: Optimization > Provide Frame Pointer

#### Architectures

-f[no-]omit-frame-pointer: IA-32 architecture, Intel® 64 architecture /Oy[-]: IA-32 architecture

## **Syntax**

Linux and Mac OS X: -fomit-frame-pointer

-fno-omit-frame-pointer

Windows: /Oy

/Oy-

## Arguments

None

### Default

-fomit-frame- EBP is used as a general-purpose register in

pointer optimizations. However, on Linux\* and Mac OS X

or /Oy systems, the default is -fno-omit-frame-

pointer if option -00 or -g is specified. On

Windows\* systems, the default is /Oy- if option

/Od is specified.

## Description

These options determine whether EBP is used as a general-purpose register in optimizations. Options -fomit-frame-pointer and /Oy allow this use.

Options -fno-omit-frame-pointer and /Oy- disallow it.

Some debuggers expect EBP to be used as a stack frame pointer, and cannot produce a stack backtrace unless this is so. The -fno-omit-frame-pointer and /Oy- options direct the compiler to generate code that maintains and uses

EBP as a stack frame pointer for all functions so that a debugger can still

produce a stack backtrace without doing the following:

• For -fno-omit-frame-pointer: turning off optimizations with -00

For /Oy-: turning off /O1, /O2, or /O3 optimizations

The -fno-omit-frame-pointer option is set when you specify option -00 or

the -g option. The -fomit-frame-pointer option is set when you specify

option -01, -02, or -03.

The /Oy option is set when you specify the /O1, /O2, or /O3 option. Option

/Oy- is set when you specify the /Od option.

Using the -fno-omit-frame-pointer or /Oy- option reduces the number of

available general-purpose registers by 1, and can result in slightly less efficient

code.

**Alternate Options** 

Linux and Mac OS X: -fp (this is a deprecated option)

Windows: None

fp-model, fp

Controls the semantics of floating-point calculations.

**IDE Equivalent** 

Windows: None

Linux: None

Mac OS X: Floating Point > Floating Point Model (precise, fast, fast=2,

strict, source)

Floating Point > Reliable Floating Point Exceptions Model (fp-model

except)

Architectures

IA-32, Intel® 64, IA-64 architectures

179

## **Syntax**

Linux and Mac OS X: -fp-modelkeyword

Windows: /fp:keyword

### **Arguments**

keyword Specifies the semantics to be used. Possible

values are:

precise Enables value-safe

optimizations on floating-point data and rounds intermediate results to source-defined

precision.

fast[=1|2] Enables more aggressive

optimizations on floating-point

data.

strict Enables precise and except,

disables contractions, and enables the property that allows modification of the floating-point environment.

source Rounds intermediate results to

source-defined precision and

enables value-safe

optimizations.

[ no- Determines whether floating-

lexcept point exception semantics are

(Linux and used.

Mac OS X)

or

except[-]
(Windows)

### Default

-fp-model The compiler uses more aggressive optimizations on fast=1 floating-point calculations, except when -00 (Linux and or Mac OS X) or /od (Windows) is specified.

/fp:fast=1

### Description

This option controls the semantics of floating-point calculations.

The *keywords* can be considered in groups:

- Group A: precise, fast, strict
- Group B: source
- Group C: except (or the negative form)

You can use more than one *keyword*. However, the following rules apply:

- You cannot specify fast and except together in the same compilation. You can specify any other combination of group A, group B, and group C.
   Since fast is the default, you must not specify except without a group A or group B keyword.
- You should specify only one keyword from group A and only one keyword from group B. If you try to specify more than one keyword from either group A or group B, the last (rightmost) one takes effect.
- If you specify except more than once, the last (rightmost) one takes effect.

Option	Description
-fp-model precise or	Tells the compiler to strictly adhere to
/fp:precise	value-safe optimizations when
	implementing floating-point calculations.
	It disables optimizations that can

Option	Description
	change the result of floating-point
	calculations. These semantics ensure
	the accuracy of floating-point
	computations, but they may slow
	performance.
	The compiler assumes the default
	floating-point environment; you are not
	allowed to modify it.
	Floating-point exception semantics are
	disabled by default. To enable these
	semantics, you must also specify -fp-
	model except or /fp:except.
	For information on the semantics used
	to interpret floating-point calculations in
	the source code, see precise in
	Floating-point Operations: Using the -
	fp-model (/fp) <b>Option</b> .
-fp-model fast[=1 2] or	Tells the compiler to use more
/fp:fast[=1 2]	aggressive optimizations when
	implementing floating-point calculations.
	These optimizations increase speed, but
	may alter the accuracy of floating-point
	computations.
	Specifying fast is the same as specifying
	fast=1. fast=2 may produce faster
	and less accurate results.
	Floating-point exception semantics are
	disabled by default and they cannot be
	enabled because you cannot specify

Option	Description
	fast and except together in the same
	compilation. To enable exception
	semantics, you must explicitly specify
	another keyword (see other keyword
	descriptions for details).
	For information on the semantics used
	to interpret floating-point calculations in
	the source code, see fast in Floating-
	point Operations: Using the -fp-model
	(/fp) Option.
-fp-model strict or /fp:strict	Tells the compiler to strictly adhere to
	value-safe optimizations when
	implementing floating-point calculations
	and enables floating-point exception
	semantics. This is the strictest floating-
	point model.
	The compiler does not assume the
	default <u>floating-point environment</u> ; you
	are allowed to modify it.
	Floating-point exception semantics can
	be disabled by explicitly specifying -fp-
	model no-except or /fp:except
	For information on the semantics used
	to interpret floating-point calculations in
	the source code, see strict in
	Floating-point Operations: Using the -
	fp-model (/fp) <b>Option</b> .
-fp-model source or /fp:source	This option causes intermediate results

Option	Description
	to be rounded to the precision defined in
	the source code. It also implies keyword
	precise unless it is overridden by a
	keyword from Group A.
	The compiler assumes the default
	floating-point environment; you are not
	allowed to modify it.
	For information on the semantics used
	to interpret floating-point calculations in
	the source code, see source in
	Floating-point Operations: Using the -
	fp-model (/fp) <b>Option</b> .

Note

This option cannot be used to change the default (source) precision for the calculation of intermediate results.

Note

On Windows and Linux operating systems on IA-32 architecture, the compiler, by default, implements floating-point (FP) arithmetic using SSE2 and SSE instructions. This can cause differences in floating-point results when compared to previous x87 implementations.

## **Alternate Options**

None

## **Example**

For examples of how to use this option, see *Floating-point Operations: Using the* -fp-model (/fp) Option.

### See Also

o compiler option (specifically O0)

Od compiler option

mp1, Qprec compiler option

The MSDN article Microsoft Visual C++ Floating-Point Optimization, which discusses concepts that apply to this option.

Floating-point Operations: Floating-Point Environment

## fp-model, fp

Controls the semantics of floating-point calculations.

### **IDE Equivalent**

Windows: None

Linux: None

Mac OS X: Floating Point > Floating Point Model (precise, fast, fast=2,

strict, source)

Floating Point > Reliable Floating Point Exceptions Model (fp-model

except)

#### Architectures

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

**Linux and Mac OS X**: -fp-model*keyword* 

Windows: /fp:keyword

## Arguments

keyword Specifies the semantics to be used. Possible

values are:

precise Enables value-safe

optimizations on floating-point data and rounds intermediate results to source-defined precision.

fast[=1|2] Enables more aggressive optimizations on floating-point data.

Enables precise and except,
disables contractions, and
enables the property that
allows modification of the
floating-point environment.

source Rounds intermediate results to source-defined precision and enables value-safe optimizations.

[no- Determines whether floating]except point exception semantics are
(Linux and used.
Mac OS X)
or
except[-]
(Windows)

#### Default

-fp-model The compiler uses more aggressive optimizations on

fast=1 floating-point calculations, except when -00 (Linux and

or Mac OS X) or /od (Windows) is specified.

/fp:fast=1

## Description

This option controls the semantics of floating-point calculations.

The *keywords* can be considered in groups:

- Group A: precise, fast, strict
- Group B: source
- Group C: except (or the negative form)

You can use more than one *keyword*. However, the following rules apply:

- You cannot specify fast and except together in the same compilation. You
  can specify any other combination of group A, group B, and group C.
   Since fast is the default, you must not specify except without a group A or
  group B keyword.
- You should specify only one keyword from group A and only one keyword from group B. If you try to specify more than one keyword from either group A or group B, the last (rightmost) one takes effect.
- If you specify except more than once, the last (rightmost) one takes effect.

Option	Description
-fp-model precise or	Tells the compiler to strictly adhere to
/fp:precise	value-safe optimizations when
	implementing floating-point calculations.
	It disables optimizations that can
	change the result of floating-point
	calculations. These semantics ensure
	the accuracy of floating-point
	computations, but they may slow
	performance.
	The compiler assumes the default
	floating-point environment; you are not
	allowed to modify it.
	Floating-point exception semantics are

Option	Description
	disabled by default. To enable these
	semantics, you must also specify -fp-
	model except or /fp:except.
	For information on the semantics used
	to interpret floating-point calculations in
	the source code, see precise in
	Floating-point Operations: Using the -
	fp-model (/fp) Option.
-fp-model fast[=1 2] or	Tells the compiler to use more
/fp:fast[=1 2]	aggressive optimizations when
	implementing floating-point calculations.
	These optimizations increase speed, but
	may alter the accuracy of floating-point
	computations.
	Specifying fast is the same as specifying
	fast=1. fast=2 may produce faster
	and less accurate results.
	Floating-point exception semantics are
	disabled by default and they cannot be
	enabled because you cannot specify
	fast and except together in the same
	compilation. To enable exception
	semantics, you must explicitly specify
	another keyword (see other keyword
	descriptions for details).
	For information on the semantics used
	to interpret floating-point calculations in
	the source code, see fast in Floating-
	point Operations: Using the -fp-model

Option	Description
	(/fp) Option.
-fp-model strict Of /fp:strict	Tells the compiler to strictly adhere to value-safe optimizations when implementing floating-point calculations and enables floating-point exception semantics. This is the strictest floating-point model.  The compiler does not assume the default floating-point environment; you are allowed to modify it.  Floating-point exception semantics can be disabled by explicitly specifying -fp-model no-except or /fp:except  For information on the semantics used to interpret floating-point calculations in the source code, see strict in Floating-point Operations: Using the -fp-model (/fp) Option.
-fp-model source Or /fp:source	This option causes intermediate results to be rounded to the precision defined in the source code. It also implies keyword precise unless it is overridden by a keyword from Group A.  The compiler assumes the default floating-point environment; you are not allowed to modify it.  For information on the semantics used to interpret floating-point calculations in

Option	Description
	the source code, see source in
	Floating-point Operations: Using the -
	fp-model (/fp) <b>Option</b> .

# Note

This option cannot be used to change the default (source) precision for the calculation of intermediate results.

# Note

On Windows and Linux operating systems on IA-32 architecture, the compiler, by default, implements floating-point (FP) arithmetic using SSE2 and SSE instructions. This can cause differences in floating-point results when compared to previous x87 implementations.

### **Alternate Options**

None

#### Example

For examples of how to use this option, see *Floating-point Operations: Using the* -fp-model (/fp) Option.

#### See Also

o compiler option (specifically O0)

od compiler option

mp1, Qprec compiler option

The MSDN article Microsoft Visual C++ Floating-Point Optimization, which discusses concepts that apply to this option.

Floating-point Operations: Floating-Point Environment

# fp-port, Qfp-port

Rounds floating-point results after floating-point operations.

## IDE Equivalent

Windows: Floating-Point > Round Floating-Point Results

Linux: None

Mac OS X: Floating-Point > Round Floating-Point Results

### **Architectures**

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: -fp-port

-no-fp-port

Windows: /Qfp-port

/Qfp-port-

### Arguments

None

#### Default

-no-fp- The default rounding behavior depends on the compiler's

port code generation decisions and the precision parameters

or/Qfp- of the operating system.

port-

## Description

This option rounds floating-point results after floating-point operations. Rounding to user-specified precision occurs at assignments and type conversions. This has some impact on speed.

The default is to keep results of floating-point operations in higher precision. This provides better performance but less consistent floating-point results.

None

# fp-relaxed, Qfp-relaxed

Enables use of faster but slightly less accurate code sequences for math functions.

## **IDE Equivalent**

None

### Architectures

IA-64 architecture

## **Syntax**

Linux: -fp-relaxed

-no-fp-relaxed

Mac OS X: None

Windows: /Qfp-relaxed

/Qfp-relaxed-

## Arguments

None

## Default

-no-fp-relaxed Default code sequences are used for math or/Qfp-relaxed functions.

## Description

This option enables use of faster but slightly less accurate code sequences for math functions, such as divide and sqrt. When compared to strict IEEE\* precision,

this option slightly reduces the accuracy of floating-point calculations performed by these functions, usually limited to the least significant digit.

This option also enables the performance of more aggressive floating-point transformations, which may affect accuracy.

## **Alternate Options**

Linux: -IPF-fp-relaxed (this is a deprecated option)

Windows: /QIPF-fp-relaxed (this is a deprecated option)

#### See Also

fp-model, fp compiler option

## fp-speculation, Qfp-speculation

Tells the compiler the mode in which to speculate on floating-point operations.

## **IDE Equivalent**

Windows: Floating Point > Floating-Point Speculation

Linux: None

Mac OS X: Floating Point > Floating-Point Speculation

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: -fp-speculation=mode

Windows: /Qfp-speculation:mode

### Arguments

mode Is the mode for floating-point operations. Possible

values are:

fast Tells the compiler to speculate

on floating-point operations.

safe Tells the compiler to disable

speculation if there is a

possibility that the speculation

may cause a floating-point

exception.

strict Tells the compiler to disable

speculation on floating-point

operations.

off This is the same as specifying

strict.

#### Default

The compiler speculates on floating-point speculation=fast operations. This is also the behavior when or/Qfp- optimizations are enabled. However, if you specify speculation:fast no optimizations (-00 on Linux; /Od on Windows), the default is -fp-speculation=safe (Linux) or /Qfp-speculation:safe (Windows).

## Description

This option tells the compiler the mode in which to speculate on floating-point operations.

## **Alternate Options**

None

#### See Also

Floating-point Operations: Floating-point Options Quick Reference

## fp-stack-check, Qfp-stack-check

Tells the compiler to generate extra code after every function call to ensure that the floating-point stack is in the expected state.

### **IDE Equivalent**

Windows: Floating-Point > Check Floating-point Stack

Linux: None

Mac OS X: Floating-Point > Check Floating-point Stack

#### **Architectures**

IA-32, Intel® 64 architectures

### Syntax

Linux and Mac OS X: -fp-stack-check

Windows: /Qfp-stack-check

### **Arguments**

None

#### Default

OFF There is no checking to ensure that the floating-point (FP) stack is in the expected state.

#### Description

This option tells the compiler to generate extra code after every function call to ensure that the floating-point (FP) stack is in the expected state.

By default, there is no checking. So when the FP stack overflows, a NaN value is put into FP calculations and the program's results differ. Unfortunately, the overflow point can be far away from the point of the actual bug. This option places code that causes an access violation exception immediately after an incorrect call occurs, thus making it easier to locate these issues.

None

### See Also

Floating-point Operations:

Checking the Floating-point Stack State

## fpconstant

Tells the compiler that single-precision constants assigned to double-precision variables should be evaluated in double precision.

## **IDE Equivalent**

Windows: Floating-Point > Extend Precision of Single-Precision Constants

Linux: None

Mac OS X: Floating-Point > Extend Precision of Single-Precision Constants

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -fpconstant

-nofpconstant

Windows: /fpconstant

/nofpconstant

## Arguments

None

### Default

nofpconstant Single-precision constants assigned to double-precision variables are evaluated in single precision according to

Fortran 95/90 Standard rules.

Description

This option tells the compiler that single-precision constants assigned to double-

precision variables should be evaluated in double precision.

This is extended precision. It does not comply with the Fortran 95/90 standard,

which requires that single-precision constants assigned to double-precision

variables be evaluated in single precision.

It allows compatibility with FORTRAN 77, where such extended precision was

allowed. If this option is not used, certain programs originally created for

FORTRAN 77 compilers may show different floating-point results because they

rely on the extended precision for single-precision constants assigned to double-

precision variables.

**Alternate Options** 

None

Example

In the following example, if you specify fpconstant, identical values are

assigned to D1 and D2. If you omit fpconstant, the compiler will obey the

Fortran 95/90 Standard and assign a less precise value to D1:

REAL (KIND=8) D1, D2

DATA D1 /2.71828182846182/ ! REAL (KIND=4) value expanded to double

DATA D2 /2.71828182846182D0/ ! Double value assigned to double

fpe

Allows some control over floating-point exception handling for the main program

at run-time.

**IDE Equivalent** 

Windows: Floating-Point > Floating-Point Exception Handling

Linux: None

197

## Mac OS X: Floating-Point > Floating-Point Exception Handling

### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -fpen

Windows: /fpe:n

### **Arguments**

Specifies the floating-point exception handling. Possible valuesare:

0

Floating-point invalid, divide-by-zero, and overflow exceptions are enabled. If any such exceptions occur, execution is aborted. This option sets the -ftz (Linux and Mac OS X) or /Qftz (Windows) option; therefore underflow results will be set to zero unless you explicitly specify -no-ftz (Linux and Mac OS X) or /Qftz- (Windows). On systems using IA-64 architecture, underflow behavior is equivalent to specifying option -ftz or /Qftz. On systems using IA-32 architecture or Intel® 64 architecture, underflow results from SSE instructions, as well as x87 instructions, will be set to zero. By contrast, option -ftz or /Qftz only sets SSE underflow results to zero.

To get more detailed location information about where the error occurred, use option traceback.

1

All floating-point exceptions are disabled. On systems using IA-64 architecture, underflow behavior is equivalent to specifying option -ftz or /Qftz. On systems using IA-32 architecture or Intel® 64 architecture, underflow results from SSE instructions, as well as x87 instructions, will be set to zero.

3

All floating-point exceptions are disabled. Floating-point underflow is gradual, unless you explicitly specify a compiler option that enables flush-to-zero, such as -ftz or /Qftz, 03, or 02 on systems using IA-32 architecture or Intel® 64 architecture. This setting provides full IEEE support.

#### Default

-fpe3 All floating-point exceptions are disabled. Floating-point
 or underflow is gradual, unless you explicitly specify a compiler
 /fpe:3 option that enables flush-to-zero.

#### Description

This option allows some control over floating-point exception handling for the main program at run-time. This includes whether exceptional floating-point values are allowed and how precisely run-time exceptions are reported.

The fpe option affects how the following conditions are handled:

• When floating-point calculations result in a divide by zero, overflow, or invalid

operation.

When floating-point calculations result in an underflow.

• When a denormalized number or other exceptional number (positive infinity,

negative infinity, or a NaN) is present in an arithmetic expression.

When enabled exceptions occur, execution is aborted and the cause of the abort

reported to the user. If compiler option traceback is specified at compile time,

detailed information about the location of the abort is also reported.

This option does not enable underflow exceptions, input denormal exceptions, or

inexact exceptions.

**Alternate Options** 

None

See Also

ftz, Qftz compiler option

traceback compiler option

Using the -fpe or /fpe Compiler Option

fpic

Determines whether the compiler generates position-independent code.

**IDE Equivalent** 

None

Architectures

IA-32, Intel® 64, IA-64 architectures

**Syntax** 

Linux and Mac OS X: -fpic

-fno-pic

Windows: None

### **Arguments**

None

#### Default

-fno-pic or On systems using IA-32 or Intel® 64 architecture, the

-fpic compiler does not generate position-independent code.

On systems using IA-64 architecture, the compiler

generates position-independent code.

## Description

This option determines whether the compiler generates position-independent code.

Option -fpic specifies full symbol preemption. Global symbol definitions as well as global symbol references get default (that is, preemptable) visibility unless explicitly specified otherwise.

Option -fno-pic is only valid on systems using IA-32 or Intel® 64 architecture.

On systems using IA-32 or Intel® 64 architecture, -fpic must be used when building shared objects.

This option can also be specified as -fpic.

### **Alternate Options**

None

## fpie

Tells the compiler to generate position-independent code. The generated code can only be linked into executables.

## **IDE Equivalent**

None

## Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux: -fpie

Mac OS X: None

Windows: None

## Arguments

None

#### Default

OFF The compiler does not generate position-

independent code for an executable-only object.

### Description

This option tells the compiler to generate position-independent code. It is similar to -fpic, but code generated by -fpie can only be linked into an executable. Because the object is linked into an executable, this option causes better optimization of some symbol references.

To ensure that run-time libraries are set up properly for the executable, you should also specify option -pie to the compiler driver on the link command line.

Option -fpie can also be specified as -fpie.

### **Alternate Options**

None

## See Also

fpic compiler option
pie compiler option

## fpp, Qfpp

Runs the Fortran preprocessor on source files before compilation.

## **IDE Equivalent**

Windows: Preprocessor > Preprocess Source File

Linux: None

Mac OS X: Preprocessor > Preprocess Source File

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: fpp[n]

fpp[="option"]

-nofpp

Windows: /fpp[n]

/fpp[:"option"]

/nofpp

/Qfpp[n]

/Qfpp[:"option"]

### Arguments

*n* Deprecated. Tells the compiler whether to run the

preprocessor or not. Possible values are:

O Tells the compiler not to run

the preprocessor.

1, 2, or Tells the compiler to run the

3 preprocessor.

option Is a Fortran preprocessor (fpp) option; for

example, "-macro=no", which disables macro

expansion. The quotes are required. For a list of

fpp options, see Fortran Preprocessor Options.

Default

nofpp The Fortran preprocessor is not run on files before compilation.

Description

This option runs the Fortran preprocessor on source files before they are compiled.

If the option is specified with no argument, the compiler runs the preprocessor.

If 0 is specified for n, it is equivalent to nofpp. Note that argument n is

deprecated.

We recommend you use option <code>Qoption,fpp,"option"</code> to pass fpp options to the Fortran preprocessor.

**Alternate Options** 

Linux and Mac OS X: -cpp

Windows: /Qcpp

See Also

Fortran Preprocessor Options

Qoption compiler option

fpscomp

Controls whether certain aspects of the run-time system and semantic language features within the compiler are compatible with Intel® Fortran or Microsoft\* Fortran PowerStation.

**IDE Equivalent** 

Windows: Compatibility > Use Filenames from Command Line

(/fpscomp:[no]filesfromcmd)

Compatibility > Use PowerStation I/O Format (/fpscomp:[no]ioformat)

204

Compatibility > Use PowerStation Portability Library (/fpscomp:[no]libs)

Compatibility > Use PowerStation List-Directed I/O Spacing

(/fpscomp:[no]ldio\_spacing)

Compatibility > Use PowerStation Logical Values

(/fpscomp:[no]logicals)

Compatibility > Use Other PowerStation Run-Time Behavior

(/fpscomp:[no]general)

Linux: None

Mac OS X: None

**Architectures** 

IA-32, Intel® 64, IA-64 architectures

**Syntax** 

Linux and Mac OS X: -fpscomp [keyword]

-nofpscomp

Windows: /fpscomp[:keyword]

/nofpscomp

Arguments

keyword Specifies the compatibility that the compiler should follow.

Possible values are:

none Specifies that no options should

be used for compatibility.

[no]filesfromcmd Determines what compatibility is

used when the OPEN statement

FILE= specifier is blank.

[no]general Determines what compatibility is

used when semantics differences

exist between Fortran

PowerStation and Intel® Fortran.

[no]ioformat Determines what compatibility is

used for list-directed formatted

and unformatted I/O.

[no]libs Determines whether the

portability library is passed to the

linker.

[no]ldio\_spacing Determines whether a blank is

inserted at run-time after a

numeric value before a character

value.

[no]logicals Determines what compatibility is

used for representation of

LOGICAL values.

all Specifies that all options should

be used for compatibility.

#### Default

fpscomp The portability library is passed to the linker.

#### Description

This option controls whether certain aspects of the run-time system and semantic language features within the compiler are compatible with Intel Fortran or Microsoft\* Fortran PowerStation.

If you experience problems when porting applications from Fortran PowerStation, specify fpscomp (or fpscomp all). When porting applications from Intel Fortran, use fpscomp none or fpscomp libs (the default).

## Option Description

fpscomp none Specifies that no options should be used for compatibility with

Fortran PowerStation. This is the same as specifying

nofpscomp.Option fpscomp none enables full Intel® Fortran

compatibility. If you omit fpscomp, the default is fpscomp

libs. You cannot use the fpscomp and vms options in the

same command.

fpscomp Specifies Fortran PowerStation behavior when the OPEN filesfromemd statement FILE= specifier is blank (FILE=' '). It causes the following actions to be taken at run-time:

- The program reads a filename from the list of arguments (if any) in the command line that invoked the program. If any of the command-line arguments contain a null string ("), the program asks the user for the corresponding filename. Each additional OPEN statement with a blank FILE= specifier reads the next command-line argument.
- If there are more nameless OPEN statements than command-line arguments, the program prompts for additional file names.
- In a QuickWin application, a File Select dialog box appears to request file names.

To prevent the run-time system from using the filename specified on the command line when the OPEN statement FILE specifier is omitted, specify fpscomp nofilesfromcmd. This allows the application of Intel Fortran defaults, such as the FORTn environment variable and the FORT.n file name (where n is the unit number).

The fpscomp filesfromcmd option affects the following Fortran features:

The OPEN statement FILE specifier

## Option

## **Description**

For example, assume a program OPENTEST contains the following statements:

OPEN(UNIT = 2, FILE = ' ')

OPEN(UNIT = 3, FILE = ' ')

OPEN(UNIT = 4, FILE = ' ')

The following command line assigns the file TEST.DAT to unit 2, prompts the user for a filename to associate with unit 3, then prompts again for a filename to associate with unit 4: opentest test.dat " "

 Implicit file open statements such as the WRITE, READ, and ENDFILE statements Unopened files referred to in READ or WRITE statements are opened implicitly as if there had been an OPEN statement with a name specified as all blanks. The name is read from the command line.

# fpscomp general

Specifies that Fortran PowerStation semantics should be used when a difference exists between Intel Fortran and Fortran PowerStation. The fpscomp general option affects the following Fortran features:

- The BACKSPACE statement:
- It allows files opened with ACCESS='APPEND' to be used with the BACKSPACE statement.
- It allows files opened with ACCESS='DIRECT' to be used with the BACKSPACE statement.

Note: Allowing files that are not opened with sequential access (such as ACCESS='DIRECT') to be used with the BACKSPACE statement violates the Fortran 95 standard and may be removed in the future.

- The READ statement:
- It causes a READ from a formatted file opened for direct

# Option Description

access to read records that have the same record type format as Fortran PowerStation. This consists of accounting for the trailing Carriage Return/Line Feed pair (<CR><LF>) that is part of the record. It allows sequential reads from a formatted file opened for direct access.

Note: Allowing files that are not opened with sequential access (such as ACCESS='DIRECT') to be used with the sequential READ statement violates the Fortran 95 standard and may be removed in the future.

- It allows the last record in a file opened with FORM='FORMATTED' and a record type of STREAM\_LF or STREAM\_CR that does not end with a proper record terminator (<line feed> or <carriage return>) to be read without producing an error.
- It allows sequential reads from an unformatted file opened for direct access.
- Note: Allowing files that are not opened with sequential access (such as ACCESS='DIRECT') to be read with the sequential READ statement violates the Fortran 95 standard and may be removed in the future.
- The INQUIRE statement:
- The CARRIAGECONTROL specifier returns the value "UNDEFINED" instead of "UNKNOWN" when the carriage control is not known.
- The NAME specifier returns the file name "UNKNOWN" instead of filling the file name with spaces when the file name is not known.
- The SEQUENTIAL specifier returns the value "YES" instead of "NO" for a direct access formatted file.

# **Option Description**

- The UNFORMATTED specifier returns the value "NO" instead of "UNKNOWN" when it is not known whether unformatted I/O can be performed to the file.
   Note: Returning the value "NO" instead of "UNKNOWN" for this specifier violates the Fortran 95 standard and may be removed in the future.
- The OPEN statement:
- If a file is opened with an unspecified STATUS keyword value, and is not named (no FILE specifier), the file is opened as a scratch file.

For example:

OPEN (UNIT = 4)

- In contrast, when fpscomp nogeneral is in effect with an unspecified STATUS value with no FILE specifier, the FORTn environment variable and the FORT.n file name are used (where n is the unit number).
- If the STATUS value was not specified and if the name of the file is "USER", the file is marked for deletion when it is closed.
- It allows a file to be opened with the APPEND and READONLY characteristics.
- If the default for the CARRIAGECONTROL specifier is assumed, it gives "LIST" carriage control to direct access formatted files instead of "NONE".
- If the default for the CARRIAGECONTROL specifier is assumed and the device type is a terminal file, the file is given the default carriage control value of "FORTRAN" instead of "LIST".
- It gives an opened file the additional default of write sharing.
- It gives the file a default block size of 1024 instead of 8192.

### Option Description

- If the default for the MODE and ACTION specifier is assumed and there was an error opening the file, try opening the file as read only, then write only.
- If a file that is being re-opened has a different file type than the current existing file, an error is returned.
- It gives direct access formatted files the same record type as Fortran PowerStation. This means accounting for the trailing Carriage Return/Line Feed pair (<CR><LF>) that is part of the record.
- The STOP statement: It writes the Fortran PowerStation output string and/or returns the same exit condition values.
- The WRITE statement:
- Writing to formatted direct files
   When writing to a formatted file opened for direct access, records are written in the same record type format as Fortran PowerStation. This consists of adding the trailing Carriage Return/Line Feed pair <CR><LF>) that is part of the record. It ignores the CARRIAGECONTROL specifier setting when writing to a formatted direct access file.
- Interpreting Fortran carriage control characters
   When interpreting Fortran carriage control characters during formatted I/O, carriage control sequences are written that are the same as Fortran PowerStation. This is true for the "Space, 0, 1 and + " characters.
- Performing non-advancing I/O to the terminal
   When performing non-advancing I/O to the terminal, output is written in the same format as Fortran PowerStation.
- Interpreting the backslash (\) and dollar (\$) edit descriptors
   When interpreting backslash and dollar edit descriptors

# **Option Description** during formatted I/O, sequences are written the same as Fortran PowerStation. Performing sequential writes It allows sequential writes from an unformatted file opened for direct access. Note: Allowing files that are not opened with sequential access (such as ACCESS='DIRECT') to be read with the sequential WRITE statement violates the Fortran 95 standard and may be removed in the future. Specifying fpscomp general sets fpscomp ldio\_spacing. fpscomp Specifies that Fortran PowerStation semantic conventions and ioformat record formats should be used for list-directed formatted and unformatted I/O. The fpscomp informat option affects the following Fortran features: • The WRITE statement: For formatted list-directed WRITE statements, formatted internal list-directed WRITE statements, and formatted namelist WRITE statements, the output line, field width values, and the list-directed data type semantics are determined according to the following sample for real constants (N below): For $1 \le N < 10**7$ , use F15.6 for single precision or F24.15 for double. For N < 1 or N >= 10\*\*7, use E15.6E2 for single precision or E24.15E3 for double. See the Fortran PowerStation documentation for more detailed information about the other data types affected. • For unformatted WRITE statements, the unformatted file

semantics are dictated according to the Fortran PowerStation

# Option Description

documentation; these semantics are different from the Intel Fortran file format. See the Fortran PowerStation documentation for more detailed information.

The following table summarizes the default output formats for list-directed output with the intrinsic data types:

Data Type	Output Format with fpscomp noioformat	Output Format with fpscomp ioformat
BYTE	15	l12
LOGICAL (all)	L2	L2
INTEGER(1)	15	l12
INTEGER(2)	17	l12
INTEGER(4)	l12	l12
INTEGER(8)	122	122
REAL(4)	1PG15.7E2	1PG16.6E2
REAL(8)	1PG24.15E3	1PG25.15E3
COMPLEX(4)	'( ',1PG14.7E2, ', ',1PG14.7E2, ') '	'( ',1PG16.6E2, ', ',1PG16.6E2, ') '
COMPLEX(8)	'( ',1PG23.15E3, ', ',1PG23.15E3, ') '	'( ',1PG25.15E3, ', ',1PG25.15E3, ') '
CHARACTER	Aw	Aw

- The READ statement:
- For formatted list-directed READ statements, formatted internal list-directed READ statements, and formatted namelist READ statements, the field width values and the listdirected semantics are dictated according to the following

Option	Description
Option	sample for real constants (N below):  For 1 <= N < 10**7, use F15.6 for single precision or F24.15 for double.  For N < 1 or N >= 10**7, use E15.6E2 for single precision or E24.15E3 for double.  See the Fortran PowerStation documentation for more detailed information about the other data types affected.  For unformatted READ statements, the unformatted file semantics are dictated according to the Fortran PowerStation documentation; these semantics are different from the Intel
	Fortran file format. See the Fortran PowerStation documentation for more detailed information.
fpscomp nolibs	Prevents the portability library from being passed to the linker.
fpscomp ldio_spacing	Specifies that at run time a blank should not be inserted after a numeric value before a character value (undelimited character string). This representation is used by Intel Fortran releases before Version 8.0 and by Fortran PowerStation. If you specify fpscomp general, it sets fpscomp ldio_spacing.
fpscomp logicals	Specifies that integers with a non-zero value are treated as true, integers with a zero value are treated as false. The literal constant .TRUE. has an integer value of 1, and the literal constant .FALSE. has an integer value of 0. This representation is used by Intel Fortran releases before Version 8.0 and by Fortran PowerStation. The default is fpscomp nologicals, which specifies that odd integer values (low bit one) are treated as true and even integer values (low bit zero) are treated as false. The literal constant .TRUE. has an integer value of -1, and

### **Option**

### **Description**

the literal constant .FALSE. has an integer value of 0. This representation is used by Compag\* Visual Fortran. The internal representation of LOGICAL values is not specified by the Fortran standard. Programs which use integer values in LOGICAL contexts, or which pass LOGICAL values to procedures written in other languages, are non-portable and may not execute correctly. Intel recommends that you avoid coding practices that depend on the internal representation of LOGICAL values. The fpscomp logical option affects the results of all logical expressions and affects the return value for the following Fortran features:

- The INQUIRE statement specifiers OPENED, IOFOCUS, EXISTS, and NAMED
- The EOF intrinsic function
- The BTEST intrinsic function
- The lexical intrinsic functions LLT, LLE, LGT, and LGE

fpscomp all Specifies that all options should be used for compatibility with Fortran PowerStation. This is the same as specifying fpscomp with no keyword. Option fpscomp all enables full compatibility with Fortran PowerStation.

#### **Alternate Options**

None

#### See Also

Building Applications: Microsoft Fortran PowerStation Compatible Files

FR

See free.

Intel® Fortran Compiler User and Reference Guides fr32 Disables the use of the high floating-point registers. **IDE Equivalent** None Architectures IA-64 architecture **Syntax** Linux: -fr32 Mac OS X: None Windows: None Arguments None Default The use of the high floating-point registers is enabled. OFF Description This option disables the use of the high floating-point registers. Only the lower 32 floating-point registers are used. **Alternate Options** None free

# Specifies source files are in free format.

IDE Equivalent

Windows: Language > Source File Format (/free, /fixed)

Linux: None

Mac OS X: Language > Source File Format (/free, /fixed)

#### Architectures

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: -free

-nofree

Windows: /free

/nofree

### Arguments

None

#### Default

OFF The source file format is determined from the file extension.

#### Description

This option specifies source files are in free format. If this option is not specified, format is determined as follows:

- Files with an extension of .f90, .F90, or .i90 are free-format source files.
- Files with an extension of .f, .for, .FOR, .ftn, or .i are fixed-format files.

### **Alternate Options**

Linux and Mac OS X: -FR

Windows: /nofixed, /FR, /4Yf

#### See Also

fixed compiler option

### fsource-asm

Produces an assembly listing with source code annotations.

### **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: -fsource-asm

Windows: None

### Arguments

None

#### Default

OFF No source code annotations appear in the assembly listing file, if one is produced.

### Description

This option produces an assembly listing file with source code annotations. The assembly listing file shows the source code as interspersed comments.

To use this option, you must also specify option -s, which causes an assembly listing to be generated.

### **Alternate Options**

Linux and Mac OS X: None

Windows: /asmattr:source, /FAs

#### See Also

### S compiler option

### fstack-security-check, GS

Determines whether the compiler generates code that detects some buffer overruns.

### **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64 architectures

#### **Syntax**

Linux and Mac OS X: -fstack-security-check

-fno-stack-security-check

Windows: /GS

/GS-

#### Arguments

None

### Default

```
-fno-stack-security- The compiler does not detect buffer check overruns.
```

or /GS-

### Description

This option determines whether the compiler generates code that detects some buffer overruns that overwrite the return address. This is a common technique for exploiting code that does not enforce buffer size restrictions.

The /GS option is supported with Microsoft Visual Studio .NET 2003\* and Microsoft Visual Studio 2005\*.

### **Alternate Options**

Linux and Mac OS X: -f[no-]stack-protector

Windows: None

## fstack-security-check, GS

Determines whether the compiler generates code that detects some buffer overruns.

### **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64 architectures

### **Syntax**

Linux and Mac OS X: -fstack-security-check

-fno-stack-security-check

Windows: /GS

/GS-

### Arguments

None

### Default

-fno-stack-security- The compiler does not detect buffer check overruns.

or /GS-

### Description

This option determines whether the compiler generates code that detects some buffer overruns that overwrite the return address. This is a common technique for exploiting code that does not enforce buffer size restrictions.

The /GS option is supported with Microsoft Visual Studio .NET 2003\* and Microsoft Visual Studio 2005\*.

### **Alternate Options**

Linux and Mac OS X: -f[no-]stack-protector

Windows: None

### fsyntax-only

See syntax-only.

### ftrapuv, Qtrapuv

Initializes stack local variables to an unusual value to aid error detection.

### **IDE Equivalent**

Windows: Data > Initialize stack variables to an unusual value

Linux: None

Mac OS X: Run-Time > Initialize Stack Variables to an Unusual Value

#### Architectures

IA-32, Intel® 64, IA-64 architectures

### Syntax

Linux and Mac OS X: -ftrapuv

Windows: /Qtrapuv

### Arguments

None

#### Default

OFF The compiler does not initialize local variables.

Description

This option initializes stack local variables to an unusual value to aid error detection. Normally, these local variables should be initialized in the application. The option sets any uninitialized local variables that are allocated on the stack to a value that is typically interpreted as a very large integer or an invalid address. References to these variables are then likely to cause run-time errors that can

help you detect coding errors.

This option sets option -q (Linux and Mac OS X) and /zi or /z7 (Windows).

Alternate Options

None

See Also

g, Zi, Z7 compiler options

ftz, Qftz

Flushes denormal results to zero.

**IDE Equivalent** 

Windows: (IA-32 and IA-64 architectures): Floating Point > Flush Denormal

Results to Zero

(Intel® 64 architecture): None

Linux: None

Mac OS X: Floating Point > Flush Denormal Results to Zero

Architectures

IA-32, Intel® 64, IA-64 architectures

Syntax

Linux and Mac OS X: -ftz

-no-ftz

Windows: /Oftz

/Qftz-

#### Arguments

None

#### Default

Systems using IA-64 architecture: - On systems using IA-64

no-ftz or /Oftz-

Systems using IA-32 architecture

and Intel® 64 architecture: -ftz or systems using IA-32 architecture

/Qftz

architecture, the compiler lets

results gradually underflow. On

and Intel® 64 architecture,

denormal results are flushed to

zero.

#### Description

This option flushes denormal results to zero when the application is in the gradual underflow mode. It may improve performance if the denormal values are not critical to your application's behavior.

This option sets or resets the FTZ and the DAZ hardware flags. If FTZ is ON, denormal results from floating-point calculations will be set to the value zero. If FTZ is OFF, denormal results remain as is. If DAZ is ON, denormal values used as input to floating-point instructions will be treated as zero. If DAZ is OFF, denormal instruction inputs remain as is. Systems using IA-64 architecture have FTZ but not DAZ. Systems using Intel® 64 architecture have both FTZ and DAZ. FTZ and DAZ are not supported on all IA-32 architectures.

When -ftz (Linux and Mac OS X) or /Oftz (Windows) is used in combination with an SSE-enabling option on systems using IA-32 architecture (for example, xN or QxN), the compiler will insert code in the main routine to set FTZ and DAZ.

When -ftz or /Qftz is used without such an option, the compiler will insert code to conditionally set FTZ/DAZ based on a run-time processor check. -no-ftz (Linux and Mac OS X) or /Qftz- (Windows) will prevent the compiler from inserting any code that might set FTZ or DAZ.

This option only has an effect when the main program is being compiled. It sets the FTZ/DAZ mode for the process. The initial thread and any threads subsequently created by that process will operate in FTZ/DAZ mode.

Options -fpe0 and -fpe1 (Linux and Mac OS X) set -ftz. Options /fpe:0 and /fpe:1 (Windows) set /Qftz.

On systems using IA-64 architecture, optimization option O3 sets -ftz and /Qftz; optimization option O2 sets -no-ftz (Linux) and /Qftz- (Windows). On systems using IA-32 architecture and Intel® 64 architecture, every optimization option O level, except O0, sets -ftz and /Qftz.

If this option produces undesirable results of the numerical behavior of your program, you can turn the FTZ/DAZ mode off by using -no-ftz or /Qftz- in the command line while still benefiting from the O3 optimizations.



Options -ftz and /Qftz are performance options. Setting these options does not guarantee that all denormals in a program are flushed to zero. They only cause denormals generated at run time to be flushed to zero.

### **Alternate Options**

None

#### See Also

x, Qx compiler option

Floating-point Operations: Using the -fpe or /fpe Compiler Option Intrinsics Reference:

### func-groups

This is a deprecated option. See <u>prof-func-groups</u>.

### funroll-loops

See unroll, Qunroll.

#### fverbose-asm

Produces an assembly listing with compiler comments, including options and version information.

### **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: -fverbose-asm

-fno-verbose-asm

Windows: None

#### **Arguments**

None

### Default

-fno-verbose- No source code annotations appear in the

asm assembly listing file, if one is produced.

### Description

This option produces an assembly listing file with compiler comments, including options and version information.

To use this option, you must also specify -S, which sets -fverbose-asm.

If you do not want this default when you specify -S, specify -fno-verbose-asm.

### **Alternate Options**

None

### See Also

s compiler option

### fvisibility

Specifies the default visibility for global symbols or the visibility for symbols in a file.

### **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

### Syntax

Linux and Mac OS X: -fvisibility=keyword

-fvisibility-keyword=file

Windows: None

### Arguments

*keyword* Specifies the visibility setting. Possible values are:

default Sets visibility to default.

extern Sets visibility to extern.

hidden Sets visibility to hidden.

internal Sets visibility to internal.

protected Sets visibility to protected.

file Is the pathname of a file containing the list of

symbols whose visibility you want to set. The

symbols must be separated by whitespace (spaces, tabs, or newlines).

#### Default

### Description

This option specifies the default visibility for global symbols (syntax – fvisibility=keyword) or the visibility for symbols in a file (syntax – fvisibility-keyword=file).

Visibility specified by -fvisibility-keyword=file overrides visibility specified by -fvisibility=keyword for symbols specified in a file.

Option	Description	
-fvisibility=default	Sets visibility of symbols to default. This	
-fvisibility-default=file	means other components can reference the symbol, and the symbol definition can be overridden (preempted) by a definition of the same name in another component.	
-fvisibility=extern -fvisibility-extern=file	Sets visibility of symbols to extern. This means the symbol is treated as though it is defined in another component. It also means that the symbol can be overridden by a definition of the same name in another component.	
-fvisibility=hidden -fvisibility-hidden=file	Sets visibility of symbols to hidden. This means that other components cannot directly reference the symbol. However, its address may be passed to other	

Option	Description
	components indirectly.
-fvisibility=internal	Sets visibility of symbols to internal. This
-fvisibility-internal=file	means the symbol cannot be referenced
	outside its defining component, either
	directly or indirectly.
-fvisibility=protected	CELL_TEXT
-fvisibility-protected=file	

If an -fvisibility option is specified more than once on the command line, the last specification takes precedence over any others.

If a symbol appears in more than one visibility *file*, the setting with the least visibility takes precedence.

The following shows the precedence of the visibility settings (from greatest to least visibility):

- extern
- default
- protected
- hidden
- internal

Note that extern visibility only applies to functions. If a variable symbol is specified as extern, it is assumed to be default.

### **Alternate Options**

None

#### Example

A file named prot.txt contains symbols a, b, c, d, and e. Consider the following:

-fvisibility-protected=prot.txt

This option sets protected visibility for all the symbols in the file. It has the same effect as specifying fvisibility=protected in the declaration for each of the symbols.

#### See Also

Optimizing Applications: Symbol Visibility Attribute Options (Linux\* and Mac OS\* X)

### g, Zi, Z7

Tells the compiler to generate full debugging information in the object file.

### **IDE Equivalent**

Windows: **General > Debug Information Format** (/Z7, /Zd, /Zi)

Linux: None

Mac OS X: General > Generate Debug Information (-g)

#### Architectures

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: -g

Windows: /Zi

/Z7

### Arguments

None

### Default

OFF No debugging information is produced in the object file.

### Description

This option tells the compiler to generate symbolic debugging information in the

object file for use by debuggers.

The compiler does not support the generation of debugging information in

assemblable files. If you specify this option, the resulting object file will contain

debugging information, but the assemblable file will not.

This option turns off O2 and makes O0 (Linux and Mac OS X) or Od (Windows)

the default unless 02 (or another 0 option) is explicitly specified in the same

command line.

On Linux systems using Intel® 64 architecture and Linux and Mac OS X systems

using IA-32 architecture, specifying the -g or -00 option sets the -fno-omit-

frame-pointer option.

For more information on Zi and Z7, see keyword full in debug (Windows\*).

### **Alternate Options**

Linux and Mac OS X: None

Windows: /debug:full (or /debug)

#### See Also

Zd compiler option

### G2, G2-p9000

Optimizes application performance for systems using IA-64 architecture.

### **IDE Equivalent**

Windows: Optimization > Optimize For Intel® Processor

Linux: None

Mac OS X: None

#### Architectures

IA-64 architecture

### **Syntax**

230

Linux and Mac OS X: None

Windows: /G2

/G2-p9000

### Arguments

None

#### Default

/G2- Performance is optimized for Dual-Core Intel® Itanium® 2 p9000 processor 9000 series.

### Description

These options optimize application performance for a particular Intel® processor or family of processors. The compiler generates code that takes advantage of features of IA-64 architecture.

Option	Description
G2	Optimizes for Intel® Itanium® 2
	processors.
G2-p9000	Optimizes for Dual-Core Intel® Itanium®
	2 processor 9000 series. This option
	affects the order of the generated
	instructions, but the generated
	instructions are limited to Intel®
	Itanium® 2 processor instructions
	unless the program specifies and
	executes intrinsics specific to the Dual-
	Core Intel® Itanium® 2 processor 9000
	series.

The resulting executable is backwards compatible and generated code is optimized for specific processors. For example, code generated with /G2-p9000

will run correctly on single-core Itanium® 2 processors, but it might not run as fast as if it had been generated using /G2.

#### **Alternate Options**

/G2 Linux: -mtune=itanium2

Mac OS X: None Windows: None

/G2-p9000 Linux: -mtune=itanium2-p9000, -

mcpu=itanium2-p9000 (-mcpu is a deprecated

option)

Mac OS X: None Windows: None

#### Example

In the following example, the compiled binary of the source program prog.f is optimized for the Dual-Core Intel® Itanium® 2 processor 9000 series by default. The same binary will also run on single-core Itanium® 2 processors (unless the program specifies and executes intrinsics specific to the Dual-Core Intel® Itanium® 2 processor 9000 series). All lines in the code example are equivalent.

```
ifort prog.f
ifort /G2-p9000 prog.f
```

In the following example, the compiled binary is optimized for single-core Itanium® 2 processors:

```
ifort /G2 prog.f
```

#### See Also

mtune compiler option

#### G5, G6, G7

Optimize application performance for systems using IA-32 architecture and Intel® 64 architecture. These are deprecated options.

#### **IDE Equivalent**

Windows: **Optimization > Optimize For Intel(R) Processor** (/GB, /G5, /G6, /G7)

Linux: None

Mac OS X: None

#### Architectures

IA-32, Intel® 64 architectures

#### **Syntax**

Linux and Mac OS X: None

Windows: /G5

/G6

/G7

### **Arguments**

None

#### Default

/G7 On systems using IA-32 architecture and Intel® 64 architecture, performance is optimized for Intel® Pentium® 4 processors, Intel® Xeon® processors, Intel® Pentium® M processors, and Intel® Pentium® 4 processors with Streaming SIMD Extensions 3 (SSE3) instruction support.

### Description

These options optimize application performance for a particular Intel® processor or family of processors. The compiler generates code that takes advantage of features of the specified processor.

### **Option Description**

- Optimizes for Intel® Pentium® and Pentium® with MMX™ technology processors.
- Optimizes for Intel® Pentium® Pro, Pentium® II and Pentium® III

### **Option Description**

processors.

Optimizes for Intel® Core™ Duo processors, Intel® Core™ Solo processors, Intel® Pentium® 4 processors, Intel® Xeon® processors based on the Intel® Core microarchitecture, Intel® Pentium® M processors, and Intel® Pentium® 4 processors with Streaming SIMD Extensions 3 (SSE3) instruction support.

On systems using Intel® 64 architecture, only option G7 is valid.

These options always generate code that is backwards compatible with Intel processors of the same architecture. For example, code generated with the G7 option runs correctly on Pentium III processors, although performance may be faster on Pentium III processors when compiled using or G6.

### **Alternate Options**

Windows: /GB (an alternate for /G6; this option is also deprecated)

Linux: None

#### Example

In the following example, the compiled binary of the source program prog.f is optimized, by default, for Intel® Pentium® 4 processors, Intel® Xeon® processors, Intel® Pentium® M processors, and Intel® Pentium® 4 processors with Streaming SIMD Extensions 3 (SSE3). The same binary will also run on Pentium, Pentium Pro, Pentium II, and Pentium III processors. All lines in the code example are equivalent.

```
ifort prog.f
ifort /G7 prog.f
```

In the following example, the compiled binary is optimized for Pentium processors and Pentium processors with MMX technology:

```
ifort /G5 prog.f
icl /G5 prog.c
```

#### See Also

### mtune compiler option

### gdwarf-2

Enables generation of debug information using the DWARF2 format.

### **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: -gdwarf-2

Windows: None

### **Arguments**

None

### Default

OFF No debug information is generated. However, if compiler option – g is specified, debug information is generated in the latest DWARF format, which is currently DWARF2.

### Description

This option enables generation of debug information using the DWARF2 format.

This is currently the default when compiler option -g is specified.

### **Alternate Options**

None

### See Also

g compiler option

Ge
Enables stack-checking for all functions.
This option has been <u>deprecated</u> .
IDE Equivalent
None
Architectures
IA-32, Intel® 64, IA-64 architectures
Syntax
Linux and Mac OS X: None
Windows: /Ge
Arguments
None
Default
OFF Stack-checking for all functions is disabled.
Description
This option enables stack-checking for all functions.
Alternate Options
None
gen-interfaces
Tells the compiler to generate an interface block for each routine in a source file

IDE Equivalent

Windows: **Diagnostics > Generate Interface Blocks** 

Linux: None

Mac OS X: Diagnostics > Generate Interface Blocks

#### Architectures

IA-32, Intel® 64, IA-64 architectures

#### **Syntax**

Linux and Mac OS X: -gen-interfaces [[no]source]

-nogen-interfaces

Windows: /gen-interfaces[:[no]source]

/nogen-interfaces

#### **Arguments**

None

#### Default

nogen- The compiler does not generate interface blocks for interfaces routines in a source file.

#### Description

This option tells the compiler to generate an interface block for each routine (that is, for each SUBROUTINE and FUNCTION statement) defined in the source file. The compiler generates two files for each routine, a .mod file and a .f90 file, and places them in the current directory or in the directory specified by the include (- I) or -module option. The .f90 file is the text of the interface block; the .mod file is the interface block compiled into binary form.

If source is specified, the compiler creates the \*\_mod.f90 as well as the \*\_mod.mod files. If nosource is specified, the compiler creates the \*\_mod.mod but not the \*\_mod.f90 files. If neither is specified, it is the same as specifying - gen-interfaces source (Linux and Mac OS X) or /gen-interfaces:source (Windows).

### **Alternate Options**

None

## global-hoist, Qglobal-hoist

Enables certain optimizations that can move memory loads to a point earlier in the program execution than where they appear in the source.

### **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: -global-hoist

-no-global-hoist

Windows: /Qglobal-hoist

/Qglobal-hoist-

### Arguments

None

### Default

-global- Certain optimizations are enabled that can move memory

hoist loads.

or/Qglobal-

hoist

### Description

This option enables certain optimizations that can move memory loads to a point earlier in the program execution than where they appear in the source. In most cases, these optimizations are safe and can improve performance.

The -no-global-hoist (Linux and Mac OS X) or /Qglobal-hoist- (Windows) option is useful for some applications, such as those that use shared or dynamically mapped memory, which can fail if a load is moved too early in the execution stream (for example, before the memory is mapped).

### **Alternate Options**

None

#### Gm

See keyword cvf in iface.

#### Gs

Disables stack-checking for routines with more than a specified number of bytes of local variables and compiler temporaries.

### **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

#### **Syntax**

Linux and Mac OS X: None

Windows: /Gs[n]

#### Arguments

*n* Is the number of bytes of local variables and

compiler temporaries.

### Default

Stack checking is disabled for routines with more than 4KB of stack space allocated.

### Description

This option disables stack-checking for routines with n or more bytes of local variables and compiler temporaries. If you do not specify n, you get the default of 4096.

### **Alternate Options**

None

### fstack-security-check, GS

Determines whether the compiler generates code that detects some buffer overruns.

### **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64 architectures

### **Syntax**

Linux and Mac OS X: -fstack-security-check

-fno-stack-security-check

Windows: /GS

/GS-

#### Arguments

None

### Default

-fno-stack-security- The compiler does not detect buffer

check

overruns.

or /GS-

Description

This option determines whether the compiler generates code that detects some

buffer overruns that overwrite the return address. This is a common technique for

exploiting code that does not enforce buffer size restrictions.

The /GS option is supported with Microsoft Visual Studio .NET 2003\* and

Microsoft Visual Studio 2005\*.

**Alternate Options** 

Linux and Mac OS X: -f[no-]stack-protector

Windows: None

Gz

See keyword stdcall in iface

heap-arrays

Puts automatic arrays and arrays created for temporary computations on the

heap instead of the stack.

**IDE Equivalent** 

Windows: Optimization > Heap Arrays

Linux: None

Mac OS X: Optimization > Heap Arrays

Architectures

IA-32, Intel® 64, IA-64 architectures

Syntax

241

Linux and Mac OS X: -heap-arrays [size]

-no-heap-arrays

Windows: /heap-arrays[:size]

/heap-arrays-

### Arguments

size Is an integer value representing the size of the arrays in kilobytes.

Any arrays known at compile-time to be larger than size are allocated on the heap instead of the stack.

#### Default

-no- The compiler puts automatic arrays and arrays created for

heap- temporary computations in temporary storage in the stack

arrays storage area.

or

/heap-

arrays-

#### Description

This option puts automatic arrays and arrays created for temporary computations on the heap instead of the stack.

If heap-arrays is specified and size is omitted, all automatic and temporary arrays are put on the heap. If 10 is specified for size, all automatic and temporary arrays larger than 10 KB are put on the heap.

### **Alternate Options**

None

#### Example

In Fortran, an automatic array gets it size from a run-time expression. For example:

Array X in the example above is affected by the heap-array option. Array Y is not.

### help

Displays all available compiler options or a category of compiler options.

### **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

#### **Syntax**

Linux and Mac OS X: -help[category]

Windows: /help[category]

### **Arguments**

category Is a category or class of options to display.

Possible values are:

advanced Displays advanced

optimization options that

allow fine tuning of compilation or allow control over advanced features of the compiler.

codegen Displays Code Generation

options.

compatibility Displays options affecting

language compatibility.

component Displays options for

component control.

data Displays options related to

interpretation of data in

programs or the storage of

data.

deprecated Displays options that have

been deprecated.

diagnostics Displays options that affect

diagnostic messages

displayed by the compiler.

float Displays options that affect

floating-point operations.

help Displays all the available

help categories.

inline Displays options that affect

inlining.

ipo Displays Interprocedural

Optimization (IPO) options

language Displays options affecting

the behavior of the compiler language

features.

link Displays linking or linker

options.

misc Displays miscellaneous

options that do not fit

within other categories.

openmp Displays OpenMP and

parallel processing

options.

opt Displays options that help

you optimize code.

output Displays options that

provide control over

compiler output.

pgo Displays Profile Guided

Optimization (PGO)

options.

preproc Displays options that affect

preprocessing operations.

reports Displays options for

optimization reports.

#### Default

OFF No list is displayed unless this compiler option is specified.

### Description

This option displays all available compiler options or a category of compiler options. If category is not specified, all available compiler options are displayed.

### **Alternate Options**

Linux and Mac OS X: None

Windows: /?

245

homeparams
------------

Tells the compiler to store parameters passed in registers to the stack.

**IDE Equivalent** 

None

Architectures

IA-32, Intel® 64, IA-64 architectures

**Syntax** 

Linux and Mac OS X: None

Windows: /homeparams

Arguments

None

Default

OFF Register parameters are not written to the stack.

### Description

This option tells the compiler to store parameters passed in registers to the stack.

**Alternate Options** 

None

L

Specifies an additional directory for the include path.

**IDE Equivalent** 

Windows: **General > Additional Include Directories** (/include)

Preprocessor > Additional Include Directories (/include)

Linux: None

Mac OS X: Preprocessor > Additional Include Directories (/include)

### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -Idir

Windows: /Idir

## Arguments

*dir* Is the directory to add to the include path.

#### Default

OFF The default include path is used.

# Description

This option specifies an additional directory for the include path, which is searched for module files referenced in USE statements and include files referenced in INCLUDE statements. To specify multiple directories on the command line, repeat the option for each directory you want to add.

For all USE statements and for those INCLUDE statements whose file name does not begin with a device or directory name, the directories are searched in this order:

- 1. The directory containing the first source file.
  - Note that if assume nosource\_include is specified, this directory will not be searched.
- 2. The current working directory where the compilation is taking place (if different from the above directory).

Any directory or directories specified using the I option. If multiple
directories are specified, they are searched in the order specified on the
command line, from left to right.

4. On Linux and Mac OS X systems, any directories indicated using environment variable FPATH. On Windows systems, any directories indicated using environment variable INCLUDE.

This option affects fpp preprocessor behavior and the USE statement.

# **Alternate Options**

Linux and Mac OS X: None

Windows: /include

## See Also

x compiler option

assume compiler option

# i-dynamic

This is a deprecated option. See <u>shared-intel</u>.

# i-static

This is a deprecated option. See static-intel.

i2, i4, i8

See integer-size.

## idirafter

Adds a directory to the second include file search path.

# **IDE Equivalent**

None

## Architectures

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -idirafterdir

Windows: None

# Arguments

*dir* Is the name of the directory to add.

#### Default

OFF Include file search paths include certain default directories.

# Description

This option adds a directory to the second include file search path (after -I).

# **Alternate Options**

None

# iface

Specifies the default calling convention and argument-passing convention for an application.

# IDE Equivalent

Windows: External Procedures > Calling Convention

(/iface:{cref|stdref|stdcall|cvf|default})

# **External Procedures > String Length Argument Passing**

(/iface:[no]mixed\_str\_len\_arg)

Linux: None

Mac OS X: None

# Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: None

Windows: /iface:keyword

# Arguments

keyword Specifies the calling convention or the argument-passing convention. Possible values are:

default Tells the compiler to use

the default calling

conventions.

cref Tells the compiler to use

calling conventions C,

REFERENCE.

cvf Tells the compiler to use

calling conventions

compatible with Compaq

Visual Fortran\*.

[no]mixed\_str\_len\_arg Determines the argument-

passing convention for

hidden-length character

arguments.

stdcall Tells the compiler to use

calling convention

STDCALL.

stdref Tells the compiler to use

calling conventions

STDCALL, REFERENCE.

# Default

/iface:default The default calling convention is used.

# Description

This option specifies the default calling convention and argument-passing convention for an application.

The aspects of calling and argument passing controlled by this option are as follows:

- The calling mechanism (C or STDCALL): On IA-32 architecture, these
  mechanisms differ in how the stack register is adjusted when a procedure call
  returns. On Intel® 64 and IA-64 architectures, the only calling mechanism
  available is C; requests for the STDCALL mechanism are ignored.
- The argument passing mechanism (by value or by reference)
- Character-length argument passing (at the end of the argument list or after the argument address)
- The case of external names (uppercase or lowercase)
- The name decoration (prefix and suffix)

You can also use the ATTRIBUTES compiler directive to modify these conventions on an individual basis. Note that the effects of the ATTRIBUTES directive do not always match that of the iface option of the same name.

Option	Description
/iface:default	Tells the compiler to use the default calling
	conventions. These conventions are as follows:
	The calling mechanism: C
	<ul> <li>The argument passing mechanism: by</li> </ul>
	reference
	<ul> <li>Character-length argument passing: at end</li> </ul>
	of argument list
	<ul> <li>The external name case: uppercase</li> </ul>
	The name decoration: Underscore prefix on

Option	Description
	IA-32 architecture, no prefix on Intel® 64 or IA-64 architecture; no suffix
/iface:cref	Tells the compiler to use the same conventions as /iface:default except that external names are lowercase.
/iface:cvf	<ul> <li>Tells the compiler to use calling conventions compatible with Compaq Visual Fortran* and Microsoft Fortran PowerStation. These conventions are as follows:</li> <li>The calling mechanism: STDCALL</li> <li>The argument passing mechanism: by reference</li> <li>Character-length argument passing: following the argument address</li> <li>The external name case: uppercase</li> <li>The name decoration: Underscore prefix on IA-32 architecture, no prefix on Intel® 64 or IA-64 architecture. On Windows* systems using IA-32 architecture, @n suffix where n is the number of bytes to be removed from the stack on exit from the procedure. No suffix on other systems.</li> </ul>
/iface:mixed_str_len_arg	Specifies argument-passing conventions for hidden-length character arguments. This option tells the compiler that the hidden length passed for a character argument is to be placed immediately after its corresponding character argument in the argument list.

Option	Description
	This is the method used by Compaq Visual Fortran*. When porting mixed-language programs that pass character arguments, either this option must be specified correctly or the order of hidden length arguments must be changed in the source code. This option can be used in addition to other /iface options.
/iface:stdcall	<ul> <li>Tells the compiler to use the following conventions:</li> <li>The calling mechanism: STDCALL</li> <li>The argument passing mechanism: by value</li> <li>Character-length argument passing: at the end of the argument list</li> <li>The external name case: uppercase</li> <li>The name decoration: Underscore prefix on IA-32 architecture, no prefix on Intel® 64 or IA-64 architecture. On Windows* systems using IA-32 architecture, @n suffix where n is the number of bytes to be removed from the stack on exit from the procedure. No suffix on other systems.</li> </ul>
/iface:stdref	Tells the compiler to use the same conventions as /iface:stdcall except that argument passing is by reference.



On Windows systems, if you specify option /iface:cref, it overrides the default for external names and causes them to be lowercase. It is as if you specified "!dec\$ attributes c, reference" for the external name.

If you specify option /iface:cref and want external names to be uppercase, you must explicitly specify option /names:uppercase.



On systems using IA-32 architecture, there must be agreement between the calling program and the called procedure as to which calling mechanism (C or STDCALL) is used or unpredictable errors may occur. If you change the default mechanism to STDCALL, you must use the ATTRIBUTES DEFAULT directive to reset the calling conventions for routines specified with the USEROPEN keyword in an OPEN statement and for comparison routines passed to the QSORT library routine.

## **Alternate Options**

/iface:cvf Linux and Mac OS X: None

Windows: /Gm

/iface:mixed\_str\_len\_arg Linux and Mac OS X: -mixed-str-

len-arg

Windows: None

/iface:nomixed\_str\_len\_arg Linux and Mac OS X: -nomixed-

str-len-arg

Windows: None

/iface:stdcall Linux and Mac OS X: None

Windows: /Gz

#### See Also

Building Applications: Programming with Mixed Languages Overview

Language Reference: ATTRIBUTES

# implicitnone

See warn.

## include

See I.

## inline

Specifies the level of inline function expansion.

# **IDE Equivalent**

None

## **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: None

Windows: /inline[:keyword]

## Arguments

*keyword* Is the level of inline function expansion. Possible values are:

none Disables inlining of user-defined

functions. This is the same as

specifying manual.

manual Disables inlining of user-defined

functions. Fortran statement functions

are always inlined.

size Enables inlining of any function.

However, the compiler decides which

functions are inlined.

This option enables interprocedural

optimizations and most speed

optimizations.

speed Enables inlining of any function. This is

the same as specifying all.

all Enables inlining of any function.

However, the compiler decides which

functions are inlined.

This option enables interprocedural

optimizations and all speed

optimizations. This is the same as

specifying inline with no *keyword*.

#### Default

OFF The compiler inlines certain functions by default.

## Description

This option specifies the level of inline function expansion.

## **Alternate Options**

inline all or Linux and Mac OS X: None

inline speed Windows: /Ob2 /Ot

inline size Linux and Mac OS X: None

Windows: /Ob2 /Os

inline manual Linux and Mac OS X: None

Windows: /Ob0

inline none Linux and Mac OS X: None

Windows: /Ob0

#### See Also

finline-functions compiler option

inline-debug-info, Qinline-debug-info

Produces enhanced source position information for inlined code. This is a

deprecated option.

**IDE Equivalent** 

None

Architectures

IA-32, Intel® 64, IA-64 architectures

Syntax

Linux and Mac OS X: -inline-debug-info

Windows:

/Qinline-debug-info

Arguments

None

Default

OFF No enhanced source position information is produced for inlined

code.

Description

This option produces enhanced source position information for inlined code. This

leads to greater accuracy when reporting the source location of any instruction. It

also provides enhanced debug information useful for function call traceback.

To use this option for debugging, you must also specify a debug enabling option.

such as -g (Linux) or /debug (Windows).

**Alternate Options** 

Linux and Mac OS X: -debug inline-debug-info

Windows: None

257

# inline-factor, Qinline-factor

Specifies the percentage multiplier that should be applied to all inlining options that define upper limits.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -inline-factor=n

-no-inline-factor

Windows: /Qinline-factor=n

/Qinline-factor-

# Arguments

*n* Is a positive integer specifying the percentage

value. The default value is 100 (a factor of 1).

# Default

-no-inline- The compiler uses default heuristics for inline

factor routine expansion.

or/Qinline-

factor-

## Description

This option specifies the percentage multiplier that should be applied to all inlining options that define upper limits:

- -inline-max-size and /Qinline-max-size
- -inline-max-total-size and /Qinline-max-total-size

- -inline-max-per-routine and /Qinline-max-per-routine
- -inline-max-per-compile and /Qinline-max-per-compile

This option takes the default value for each of the above options and multiplies it by n divided by 100. For example, if 200 is specified, all inlining options that define upper limits are multiplied by a factor of 2. This option is useful if you do not want to individually increase each option limit.

If you specify -no-inline-factor (Linux and Mac OS X) or /Qinline-factor- (Windows), the following occurs:

- Every function is considered to be a small or medium function; there are no large functions.
- There is no limit to the size a routine may grow when inline expansion is performed.
- There is no limit to the number of times some routine may be inlined into a particular routine.
- There is no limit to the number of times inlining can be applied to a compilation unit.

To see compiler values for important inlining limits, specify compiler option - opt-report (Linux and Mac OS X) or /Qopt-report (Windows).



When you use this option to increase default limits, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

#### **Alternate Options**

None

#### See Also

<u>inline-max-size</u>, <u>Qinline-max-size</u> compiler option <u>inline-max-total-size</u>, <u>Qinline-max-total-size</u> compiler option inline-max-per-routine, <u>Qinline-max-per-routine</u> compiler option

<u>inline-max-per-compile, Qinline-max-per-compile</u> compiler option opt-report, Qopt-report compiler option

Optimizing Applications:

Developer Directed Inline Expansion of User Functions

Compiler Directed Inline Expansion of User Functions

# inline-forceinline, Qinline-forceinline

Specifies that an inline routine should be inlined whenever the compiler can do so.

# **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -inline-forceinline

Windows: /Oinline-forceinline

### Default

OFF The compiler uses default heuristics for inline routine expansion.

## Description

This option specifies that a inline routine should be inlined whenever the compiler can do so. This causes the routines marked with an inline keyword or directive to be treated as if they were "forceinline".



Because C++ member functions whose definitions are included in the class declaration are considered inline functions by default, using this option will also make these member functions "forceinline" functions.

The "forceinline" condition can also be specified by using the directive cDEC\$ ATTRIBUTES FORCEINLINE.

To see compiler values for important inlining limits, specify compiler option – opt-report (Linux and Mac OS) or /Qopt-report (Windows).



When you use this option to change the meaning of inline to "forceinline", the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

# **Alternate Options**

None

#### See Also

opt-report, Qopt-report compiler option

Optimizing Applications:

Developer Directed Inline Expansion of User Functions

Compiler Directed Inline Expansion of User Functions

# inline-level, Ob

Specifies the level of inline function expansion.

# **IDE Equivalent**

Windows: **Optimization > Inline Function Expansion** 

Linux: None

Mac OS X: Optimization > Inline Function Expansion

#### Architectures

IA-32, Intel® 64, IA-64 architectures

#### **Syntax**

Linux and Mac OS X: -inline-level=n

Windows: /Obn

# Arguments

*n* Is the inline function expansion level. Possible

values are 0, 1, and 2.

## Default

-inline- This is the default if option O2 is specified or is in effect by level=2 or default. On Windows systems, this is also the default if option O3 is specified.

-inline- This is the default if option -O0 (Linux and Mac OS) or level=0 or /Od (Windows) is specified.

# Description

This option specifies the level of inline function expansion. Inlining procedures can greatly improve the run-time performance of certain programs.

Option	Description
-inline-level=0	Disables inlining of user-defined functions. Note that
or Ob0	statement functions are always inlined.
-inline-level=1	Enables inlining when an inline keyword or an inline
or Obl	directive is specified.
-inline-level=2	Enables inlining of any function at the compiler's
or Ob2	discretion.

## **Alternate Options**

Linux: -Ob (this is a deprecated option)

Mac OS X: None

Windows: None

## See Also

inline compiler option

# inline-max-per-compile, Qinline-max-per-compile

Specifies the maximum number of times inlining may be applied to an entire compilation unit.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -inline-max-per-compile=n

-no-inline-max-per-compile

Windows: /Qinline-max-per-compile=n

/Qinline-max-per-compile-

## Arguments

*n* Is a positive integer that specifies the number of

times inlining may be applied.

## Default

-no-inline-max-per- The

The compiler uses default heuristics for

inline routine expansion.

or/Qinline-max-per-

compile-

compile

# Description

This option the maximum number of times inlining may be applied to an entire compilation unit. It limits the number of times that inlining can be applied. For compilations using Interprocedural Optimizations (IPO), the entire compilation is a compilation unit. For other compilations, a compilation unit is a file.

If you specify -no-inline-max-per-compile (Linux and Mac OS X) or /Qinline-max-per-compile- (Windows), there is no limit to the number of times inlining may be applied to a compilation unit.

To see compiler values for important inlining limits, specify compiler option – opt-report (Linux and Mac OS X) or /Qopt-report (Windows).



When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

## **Alternate Options**

None

#### See Also

<u>inline-factor</u>, <u>Qinline-factor</u> compiler option <u>opt-report</u>, <u>Qopt-report</u> compiler option

Optimizing Applications:

Developer Directed Inline Expansion of User Functions
Compiler Directed Inline Expansion of User Functions

# inline-max-per-routine, Qinline-max-per-routine

Specifies the maximum number of times the inliner may inline into a particular routine.

## **IDE Equivalent**

## None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

**Linux and Mac OS X**: -inline-max-per-routine=*n* 

-no-inline-max-per-routine

Windows: /Oinline-max-per-routine=n

/Qinline-max-per-routine-

## Arguments

*n* Is a positive integer that specifies the maximum

number of times the inliner may inline into a

particular routine.

### Default

 $\hbox{-no-inline-max-per-} \qquad \hbox{The compiler uses default heuristics for} \\$ 

routine inline routine expansion.

or/Qinline-max-per-

routine-

# Description

This option specifies the maximum number of times the inliner may inline into a particular routine. It limits the number of times that inlining can be applied to any routine.

If you specify -no-inline-max-per-routine (Linux and Mac OS X) or /Qinline-max-per-routine- (Windows), there is no limit to the number of times some routine may be inlined into a particular routine.

To see compiler values for important inlining limits, specify compiler option – opt-report (Linux and Mac OS X) or /Qopt-report (Windows).

To see compiler values for important inlining limits, specify compiler option – opt-report (Linux and Mac OS X) or /Qopt-report (Windows).



When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

# **Alternate Options**

None

#### See Also

<u>inline-factor</u>, <u>Qinline-factor</u> compiler option <u>opt-report</u>, <u>Qopt-report</u> compiler option

Optimizing Applications:

Developer Directed Inline Expansion of User Functions

Compiler Directed Inline Expansion of User Functions

# inline-max-size, Qinline-max-size

Specifies the lower limit for the size of what the inliner considers to be a large routine.

# **IDE Equivalent**

None

# Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -inline-max-size=n

-no-inline-max-size

Windows: /Qinline-max-size=n

/Qinline-max-size-

## Arguments

n

Is a positive integer that specifies the minimum size of what the inliner considers to be a large routine.

### Default

-no-inline-max-size The compiler uses default heuristics for or/Qinline-max-size inline routine expansion.

## Description

This option specifies the lower limit for the size of what the inliner considers to be a large routine (a function or subroutine). The inliner classifies routines as small, medium, or large. This option specifies the boundary between what the inliner considers to be medium and large-size routines.

The inliner prefers to inline small routines. It has a preference against inlining large routines. So, any large routine is highly unlikely to be inlined.

If you specify -no-inline-max-size (Linux and Mac OS X) or /Qinline-max-size- (Windows), there are no large routines. Every routine is either a small or medium routine.

To see compiler values for important inlining limits, specify compiler option – opt-report (Linux and Mac OS X) or /Qopt-report (Windows).

To see compiler values for important inlining limits, specify compiler option - opt-report (Linux and Mac OS X) or /Qopt-report (Windows).



When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

# **Alternate Options**

None

## See Also

<u>inline-min-size</u>, <u>Qinline-min-size</u> compiler option <u>inline-factor</u>, <u>Qinline-factor</u> compiler option <u>opt-report</u>, <u>Qopt-report</u> compiler option

Optimizing Applications:

Developer Directed Inline Expansion of User Functions

Compiler Directed Inline Expansion of User Functions

# inline-max-total-size, Qinline-max-total-size

Specifies how much larger a routine can normally grow when inline expansion is performed.

# **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -inline-max-total-size=n

-no-inline-max-total-size

Windows: /Qinline-max-total-size=n

/Qinline-max-total-size-

## Arguments

*n* Is a positive integer that specifies the permitted

increase in the routine's size when inline

expansion is performed.

#### Default

```
-no-inline-max-total- The compiler uses default heuristics for size inline routine expansion.

Or/Qinline-max-total-
size-
```

## Description

This option specifies how much larger a routine can normally grow when inline expansion is performed. It limits the potential size of the routine. For example, if 2000 is specified for n, the size of any routine will normally not increase by more than 2000.

If you specify -no-inline-max-total-size (Linux and Mac OS X) or /Qinline-max-total-size- (Windows), there is no limit to the size a routine may grow when inline expansion is performed.

To see compiler values for important inlining limits, specify compiler option – opt-report (Linux and Mac OS X) or /Qopt-report (Windows).

To see compiler values for important inlining limits, specify compiler option – opt-report (Linux and Mac OS X) or /Qopt-report (Windows).



When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

# **Alternate Options**

None

#### See Also

```
<u>inline-factor</u>, <u>Qinline-factor</u> compiler option

<u>opt-report</u>, <u>Qopt-report</u> compiler option

Optimizing Applications:
```

Developer Directed Inline Expansion of User Functions

Compiler Directed Inline Expansion of User Functions

# inline-min-size, Qinline-min-size

Specifies the upper limit for the size of what the inliner considers to be a small routine.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -inline-min-size=n

-no-inline-min-size

Windows: /Qinline-min-size=n

/Qinline-min-size-

## Arguments

*n* Is a positive integer that specifies the maximum

size of what the inliner considers to be a small

routine.

#### Default

-no-inline-min-size The compiler uses default heuristics for

or/Qinline-min-size- inline routine expansion.

# Description

This option specifies the upper limit for the size of what the inliner considers to be a small routine (a function or subroutine). The inliner classifies routines as small,

medium, or large. This option specifies the boundary between what the inliner considers to be small and medium-size routines.

The inliner has a preference to inline small routines. So, when a routine is smaller than or equal to the specified size, it is very likely to be inlined. If you specify -no-inline-min-size (Linux and Mac OS X) or /Qinline-min-size (Windows), there is no limit to the size of small routines. Every routine is a small routine; there are no medium or large routines.

To see compiler values for important inlining limits, specify compiler option – opt-report (Linux and Mac OS X) or /Qopt-report (Windows).

To see compiler values for important inlining limits, specify compiler option – opt-report (Linux and Mac OS X) or /Qopt-report (Windows).



When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

## **Alternate Options**

None

## See Also

<u>inline-min-size</u>, <u>Qinline-min-size</u> compiler option <u>opt-report</u>, <u>Qopt-report</u> compiler option

Optimizing Applications:

Developer Directed Inline Expansion of User Functions
Compiler Directed Inline Expansion of User Functions

#### intconstant

Tells the compiler to use FORTRAN 77 semantics to determine the kind parameter for integer constants.

## **IDE Equivalent**

Windows: Compatibility > Use F77 Integer Constants

Linux: None

Mac OS X: Compatibility > Use F77 Integer Constants

## Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -intconstant

-nointconstant

Windows: /intconstant

/nointconstant

# Arguments

None

### Default

nointconstant The compiler uses the Fortran 95/90 default INTEGER type.

## Description

This option tells the compiler to use FORTRAN 77 semantics to determine the kind parameter for integer constants.

With FORTRAN 77 semantics, the kind is determined by the value of the constant. All constants are kept internally by the compiler in the highest precision possible. For example, if you specify option intconstant, the compiler stores an integer constant of 14 internally as INTEGER(KIND=8) and converts the constant upon reference to the corresponding proper size. Fortran 95/90 specifies that integer constants with no explicit KIND are kept internally in the default INTEGER kind (KIND=4 by default).

Note that the internal precision for floating-point constants is controlled by option fpconstant.

# **Alternate Options**

None

# integer-size

Specifies the default KIND for integer and logical variables.

## **IDE Equivalent**

Windows: **Data > Default Integer KIND** 

Linux: None

Mac OS X: Data > Default Integer KIND

# Architectures

IA-32, Intel® 64, IA-64 architectures

# Syntax

```
Linux and Mac OS X: -integer-size size
Windows: /integer-size:size
```

## Arguments

```
size Is the size for integer and logical variables. Possible values are: 16, 32, or 64.
```

#### Default

```
integer- Integer and logical variables are 4 bytes long
size 32 (INTEGER(KIND=4) and LOGICAL(KIND=4)).
```

# Description

This option specifies the default size (in bits) for integer and logical variables.

Option	Description
integer-	Makes default integer and logical variables 2 bytes long.
size 16	INTEGER and LOGICAL declarations are treated as (KIND=2).
integer-	Makes default integer and logical variables 4 bytes long.
size 32	INTEGER and LOGICAL declarations are treated as (KIND=4).
integer-	Makes default integer and logical variables 8 bytes long.
size 64	INTEGER and LOGICAL declarations are treated as (KIND=8).

# **Alternate Options**

# ip, Qip

Determines whether additional interprocedural optimizations for single-file compilation are enabled.

# **IDE Equivalent**

Windows: **Optimization > Interprocedural Optimization** 

Linux: None

Mac OS X: None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -ip

-no-ip

Windows: /Qip

/Qip-

### Arguments

None

## Default

OFF Some limited interprocedural optimizations occur, including inline function expansion for calls to functions defined within the current source file. These optimizations are a subset of full intra-file interprocedural optimizations. Note that this setting is not the same as -no-ip (Linux and Mac OS X) or /Qip- (Windows).

# Description

This option determines whether additional interprocedural optimizations for single-file compilation are enabled.

Options -ip (Linux and Mac OS X) and /Qip (Windows) enable additional interprocedural optimizations for single-file compilation.

Options -no-ip (Linux and Mac OS X) and /Qip- (Windows) may not disable inlining. To ensure that inlining of user-defined functions is disabled, specify -inline-level=0or-fno-inline (Linux and Mac OS X), or specify /Ob0 (Windows).

## **Alternate Options**

None

### See Also

finline-functions compiler option

# ip-no-inlining, Qip-no-inlining

Disables full and partial inlining enabled by interprocedural optimization options.

# **IDE Equivalent**

None

## Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -ip-no-inlining

Windows: /Qip-no-inlining

## Arguments

None

# Default

OFF Inlining enabled by interprocedural optimization options is performed.

## Description

This option disables full and partial inlining enabled by the following interprocedural optimization options:

- On Linux and Mac OS X systems: -ip or -ipo
- On Windows systems: /Qip, /Qipo, or /Ob2

It has no effect on other interprocedural optimizations.

On Windows systems, this option also has no effect on user-directed inlining specified by option /0b1.

# **Alternate Options**

None

# ip-no-pinlining, Qip-no-pinlining

Disables partial inlining enabled by interprocedural optimization options.

# **IDE Equivalent**

None

## Architectures

IA-32, Intel® 64 architectures

# **Syntax**

Linux and Mac OS X: -ip-no-pinlining

Windows: /Qip-no-pinlining

## Arguments

None

# Default

OFF Inlining enabled by interprocedural optimization options is performed.

## Description

This option disables partial inlining enabled by the following interprocedural optimization options:

- On Linux and Mac OS X systems: -ip or -ipo
- On Windows systems: /Qip or /Qipo

It has no effect on other interprocedural optimizations.

# **Alternate Options**

None

# IPF-flt-eval-method0, QIPF-flt-eval-method0

Tells the compiler to evaluate the expressions involving floating-point operands in the precision indicated by the variable types declared in the program. This is a deprecated option.

# **IDE Equivalent**

None

#### Architectures

IA-64 architecture

# **Syntax**

Linux: -IPF-flt-eval-method0

Mac OS X: None

Windows: /QIPF-flt-eval-method0

## Arguments

None

## Default

OFF Expressions involving floating-point operands are evaluated by default rules.

## Description

This option tells the compiler to evaluate the expressions involving floating-point operands in the precision indicated by the variable types declared in the program. By default, intermediate floating-point expressions are maintained in higher precision.

The recommended method to control the semantics of floating-point calculations is to use option -fp-model (Linux) or /fp (Windows).

Instead of using -IPF-flt-eval-method0 (Linux) or /QIPF-flt-evalmethod0 (Windows), you can use -fp-model source (Linux) or /fp:source
(Windows).

# **Alternate Options**

None

## See Also

fp-model, fp compiler option

# IPF-fltacc, QIPF-fltacc

Disables optimizations that affect floating-point accuracy. This is a deprecated option.

# **IDE Equivalent**

Windows: Floating Point > Floating-Point Accuracy

Linux: None

Mac OS X: None

# **Architectures**

IA-64 architecture

# **Syntax**

Linux: -IPF-fltacc

-no-IPF-fltacc

Mac OS X: None

Windows: /QIPF-fltacc

/QIPF-fltacc-

# Arguments

None

# Default

```
-no-IPF-fltacc Optimizations are enabled that affect floating-point or/QIPF-fltacc- accuracy.
```

# Description

This option disables optimizations that affect floating-point accuracy.

If the default setting is used, the compiler may apply optimizations that reduce floating-point accuracy.

You can use this option to improve floating-point accuracy, but at the cost of disabling some optimizations.

The recommended method to control the semantics of floating-point calculations is to use option -fp-model (Linux) or /fp (Windows).

Instead of using -IPF-fltacc (Linux) or /QIPF-fltacc (Windows), you can use -fp-model precise (Linux) or /fp:precise (Windows).

Instead of using -no-IPF-fltacc (Linux) or /QIPF-fltacc- (Windows), you can use -fp-model fast (Linux) or /fp:fast (Windows).

# **Alternate Options**

None

#### See Also

fp-model, fp compiler option

IPF-fma, QIPF-fma

See fma, Qfma.

# IPF-fp-relaxed, QIPF-fp-relaxed

See fp-relaxed, Qfp-relaxed.

# ipo, Qipo

Enables interprocedural optimization between files.

## **IDE Equivalent**

Windows: **Optimization > Interprocedural Optimization** 

General > Whole Program Optimization

Linux: None

Mac OS X: None

### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -ipo[n]

Windows: /Qipo[n]

## Arguments

*n* Is an optional integer that specifies the number of

object files the compiler should create. The integer

must be greater than or equal to 0.

## Default

OFF Multifile interprocedural optimization is not enabled.

# Description

This option enables interprocedural optimization between files. This is also called multifile interprocedural optimization (multifile IPO) or Whole Program Optimization (WPO).

When you specify this option, the compiler performs inline function expansion for calls to functions defined in separate files.

You cannot specify the names for the files that are created.

If n is 0, the compiler decides whether to create one or more object files based on an estimate of the size of the application. It generates one object file for small applications, and two or more object files for large applications.

If n is greater than 0, the compiler generates n object files, unless n exceeds the number of source files (m), in which case the compiler generates only m object files.

If you do not specify n, the default is 0.

# **Alternate Options**

None

## See Also

Optimizing Applications:

Interprocedural Optimization (IPO) Quick Reference Interprocedural Optimization (IPO) Overview Using IPO

# ipo-c, Qipo-c

Tells the compiler to optimize across multiple files and generate a single object file.

# **IDE Equivalent**

None

## **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -ipo-c

Windows: /Qipo-c

# Arguments

None

## Default

OFF The compiler does not generate a multifile object file.

Description

This option tells the compiler to optimize across multiple files and generate a single object file (named ipo\_out.o on Linux and Mac OS X systems; ipo\_out.obj

on Windows systems).

It performs the same optimizations as -ipo (Linux and Mac OS X) or /Qipo (Windows), but compilation stops before the final link stage, leaving an optimized object file that can be used in further link steps.

**Alternate Options** 

None

See Also

ipo, Qipo compiler option

ipo-jobs, Qipo-jobs

Specifies the number of commands (jobs) to be executed simultaneously during the link phase of Interprocedural Optimization (IPO).

**IDE Equivalent** 

None

Architectures

IA-32, Intel® 64, IA-64 architectures

**Syntax** 

Linux and Mac OS X: -ipo-jobsn

Windows: /Qipo-jobs:n

Arguments

*n* Is the number of commands (jobs) to run

simultaneously. The number must be greater than

or equal to 1.

#### Default

-ipo-jobs1 One command (job) is executed in an

or/Qipo-jobs:1 interprocedural optimization parallel build.

### Description

This option specifies the number of commands (jobs) to be executed simultaneously during the link phase of Interprocedural Optimization (IPO). It should only be used if the link-time compilation is generating more than one object. In this case, each object is generated by a separate compilation, which can be done in parallel.

This option can be affected by the following compiler options:

- -ipo (Linux and Mac OS X) or /Qipo (Windows) when applications are large enough that the compiler decides to generate multiple object files.
- -ipon (Linux and Mac OS X) or /Qipon (Windows) when n is greater than 1.
- -ipo-separate (Linux) or /Qipo-separate (Windows)



Be careful when using this option. On a multi-processor system with lots of memory, it can speed application build time. However, if n is greater than the number of processors, or if there is not enough memory to avoid thrashing, this option can increase application build time.

#### **Alternate Options**

None

#### See Also

ipo, Qipo compiler option

ipo-separate, Qipo-separate compiler option

## ipo-S, Qipo-S

Tells the compiler to optimize across multiple files and generate a single assembly file.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

### Syntax

Linux and Mac OS X: -ipo-S

Windows: /Qipo-S

## Arguments

None

### Default

OFF The compiler does not generate a multifile assembly file.

## Description

This option tells the compiler to optimize across multiple files and generate a single assembly file (named ipo\_out.s on Linux and Mac OS X systems; ipo\_out.asm on Windows systems).

It performs the same optimizations as -ipo (Linux and Mac OS X) or /Qipo (Windows), but compilation stops before the final link stage, leaving an optimized assembly file that can be used in further link steps.

## **Alternate Options**

None

## See Also

ipo, Qipo compiler option

## ipo-separate, Qipo-separate

Tells the compiler to generate one object file for every source file.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux: -ipo-separate

Mac OS X: None

Windows: /Qipo-separate

## Arguments

None

#### Default

OFF The compiler decides whether to create one or more object files.

## Description

This option tells the compiler to generate one object file for every source file. It overrides any -ipo (Linux) or /Qipo (Windows) specification.

## **Alternate Options**

None

### See Also

ipo, Qipo compiler option

## isystem

Specifies a directory to add to the start of the system include path.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -isystemdir

Windows: None

## Arguments

dir ls the directory to add to the system include path.

#### Default

OFF The default system include path is used.

## Description

This option specifies a directory to add to the system include path. The compiler searches the specified directory for include files after it searches all directories specified by the -I compiler option but before it searches the standard system directories. This option is provided for compatibility with gcc.

## **Alternate Options**

None

## ivdep-parallel, Qivdep-parallel

Tells the compiler that there is no loop-carried memory dependency in the loop following an IVDEP directive.

## **IDE Equivalent**

Windows: Optimization > IVDEP Directive Memory Dependency

Linux: None

Mac OS X: None

#### **Architectures**

IA-64 architecture

## **Syntax**

Linux: -ivdep-parallel

Mac OS X: None

Windows: /Qivdep-parallel

## Arguments

None

#### Default

OFF There may be loop-carried memory dependency in a loop that follows an IVDEP directive.

## Description

This option tells the compiler that there is no loop-carried memory dependency in the loop following an IVDEP There may be loop-carried memory dependency in a loop that follows an IVDEP directive.

This has the same effect as specifying the IVDEP:LOOP directive.

#### **Alternate Options**

None

#### See Also

Optimizing Applications: Absence of Loop-carried Memory Dependency with IVDEP Directive

ı

Tells the linker to search for a specified library when linking.

#### **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -lstring

Windows: None

## Arguments

string Specifies the library (libstring) that the linker should

search.

#### Default

OFF The linker searches for standard libraries in standard directories.

## Description

This option tells the linker to search for a specified library when linking.

When resolving references, the linker normally searches for libraries in several standard directories, in directories specified by the  $\mathbb{L}$  option, then in the library specified by the  $\mathbb{L}$  option.

The linker searches and processes libraries and object files in the order they are specified. So, you should specify this option following the last object file it applies to.

Intel® Fortran Compiler User and Reference Guides **Alternate Options** None See Also L compiler option L Tells the linker to search for libraries in a specified directory before searching the standard directories. **IDE Equivalent** None **Architectures** IA-32, Intel® 64, IA-64 architectures **Syntax** Linux and Mac OS X: -Ldir Windows: None Arguments dir Is the name of the directory to search for libraries. Default OFF The linker searches the standard directories for libraries. Description

This option tells the linker to search for libraries in a specified directory before searching for them in the standard directories.

## **Alternate Options**

None

290

#### See Also

1 compiler option

#### LD

See dll.

#### libdir

Controls whether linker options for search libraries are included in object files generated by the compiler.

## **IDE Equivalent**

Windows: Libraries > Disable Default Library Search Rules

(/libdir:[no]automatic)

**Libraries > Disable OBJCOMMENT Library Name in Object** 

(/libdir:[no]user)

Linux: None

Mac OS X: None

Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: None

Windows: /libdir[:keyword]

/nolibdir

#### Arguments

*keyword* Specifies the linker search options. Possible values are:

none Prevents any linker search options

from being included into the object

file. This is the same as specifying /nolibdir.

[no]automatic Determines whether linker search
options for libraries automatically
determined by the ifort command
driver (default libraries) are included
in the object file.

Determines whether linker search
options for libraries specified by the
OBJCOMMENT source directives are
included in the object file.

all Causes linker search options for the following libraries:

- Libraries automatically determined by the ifort command driver (default libraries)
- Libraries specified by the OBJCOMMENT directive to be included in the object file

This is the same as specifying /libdir.

### Default

/libdir:all Linker search options for libraries automatically

determined by the ifort command driver (default

libraries) and libraries specified by the OBJCOMMENT

directive are included in the object file.

## Description

This option controls whether linker options for search libraries

( /DEFAULTLIB:library) are included in object files generated by the compiler.

The linker option /DEFAULTLIB:library adds one library to the list of libraries that the linker searches when resolving references. A library specified with /DEFAULTLIB:library is searched after libraries specified on the command line and before default libraries named in .obj files.

### **Alternate Options**

/libdir:none Linux and Mac OS X: None

Windows: /Zl

### libs

Tells the compiler which type of run-time library to link to.

## **IDE Equivalent**

Windows: Libraries > Runtime Library (/libs: {static|dll|qwin|qwins},

/threads, /dbglibs)

Linux: None

Mac OS X: None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: None

Windows: /libs[:keyword]

## Arguments

keyword Specifies the type of run-time library to link to. Possible values

are:

static	Specifies a single-threaded, static
	library (same as specifying /libs).
dll	Specifies a single-threaded, dynamic-link (DLL) library.
qwin	Specifies the Fortran QuickWin library.
qwins	Specifies the Fortran Standard
	Graphics library.

#### Default

/libs:static The compiler links to a single-threaded, static run-time or /libs library.

### Description

This option tells the compiler which type of run-time library to link to.

The library can be statically or dynamically loaded, multithreaded (/threads) or single-threaded, or debug (/dbglibs) or nondebug.

If you use the /libs:dll option and an unresolved reference is found in the DLL, it gets resolved when the program is executed, during program loading, reducing executable program size.

If you use the /libs:qwin or /libs:qwins option with the /dll option, the compiler issues a warning.

You cannot use the /libs:qwin option and options /libs:dll /threads.

The following table shows which options to specify for different run-time libraries:

Type of Library	Options Required	Alternate Option
Single-threaded, static	/libs:static	/ML
	or	
	/libs or	

Type of Library	Options Required	Alternate Option
	/static	
Multithreaded	/libs:static /threads	/MT
Debug single-threaded	/libs:static /dbglibs	/MLd
Debug multithreaded	/libs:static /threads /dbglibs	/MTd
Single-threaded, dynamic-link libraries (DLLs)	/libs:dll	/MDs
Debug single-threaded, dynamic-link libraries (DLLs)	/libs:dll /dbglibs	/MDsd
Multithreaded DLLs	/libs:dll /threads	/MD
Multithreaded debug DLLs	/libs:dll /threads /dbglibs	/MDd
Fortran QuickWin multi-doc applications	/libs:qwin	/MW
Fortran standard graphics (QuickWin single-doc) applications	/libs:qwins	/MWs
Debug Fortran QuickWin multi-doc applications	/libs:qwin /dbglibs	None
Debug Fortran standard graphics (QuickWin single-doc) applications	/libs:qwins /dbglibs	None

**Alternate Options** 

/libs:dll	Linux and Mac OS X:None Windows: /MDs
/libs:static	Linux and Mac OS X: None Windows: /ML
/libs:qwin	Linux and Mac OS X: None Windows: /MW
/libs:qwins	Linux and Mac OS X: None Windows: /MWs
threads compiler op dbglibs compiler op Building Applications: Languages Overview	
link	
Passes user-specific	ed options directly to the linker at compile time.
IDE Equivalent	
None	
Architectures	
IA-32, Intel® 64, IA-	64 architectures
Syntax	
Linux and Mac OS >	(: None
Windows:	/link
Arguments	
None	
Default	

OFF No user-specified options are passed directly to the linker.

## Description

This option passes user-specified options directly to the linker at compile time.

All options that appear following /link are passed directly to the linker.

## **Alternate Options**

None

### See Also

Xlinker compiler option

## logo

Displays the compiler version information.

## **IDE Equivalent**

Windows: **General > Suppress Startup Banner** (/nologo)

Linux: None

Mac OS X: **General > Show Startup Banner** (-V)

### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -logo

-nologo

Windows: /logo

/nologo

## Arguments

None

### Default

Linux and Mac OS X: The compiler version information is not displayed.

nologo

Windows: logo The compiler version information is displayed.

## Description

This option displays the startup banner, which contains the following compiler version information:

- ID: unique identification number for the compiler
- x.y.z: version of the compiler
- · years: years for which the software is copyrighted

This option can be placed anywhere on the command line.

## **Alternate Options**

Linux and Mac OS X: -V

Windows: None

## lowercase, Qlowercase

See <u>names</u>.

#### m

Tells the compiler to generate optimized code specialized for the processor that executes your program.

### **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64 architectures

## Syntax

Linux and Mac OS X: -m[processor]

Windows: None

### Arguments

processor Indicates the processor for which code is

generated. Possible values are:

ia32 Generates code that will run on

any Pentium or later processor.

Disables any default extended

instruction settings, and any

previously set extended

instruction settings. This value

is only available on Linux

systems using IA-32

architecture.

sse This is the same as specifying

ia32.

sse2 Generates code for Intel®

Streaming SIMD Extensions 2

(Intel® SSE2). This value is

only available on Linux

systems.

sse3 Generates code for Intel®

Streaming SIMD Extensions 3

(Intel® SSE3).

ssse3 Generates code for Intel®

Supplemental Streaming SIMD

Extensions 3 (Intel® SSSE3).

sse4.1 Generates code for Intel®

Streaming SIMD Extensions 4
Vectorizing Compiler and
Media Accelerators.

#### Default

Linux For more information on the default values, see Arguments

systems: - above.

msse2

Mac OS X

systems

using IA-32

architecture:

-msse3

Mac OS X

systems

using Intel®

64

architecture:

-mssse3

#### Description

This option tells the compiler to generate optimized code specialized for the processor that executes your program.

Code generated with the values ia32, sse, sse2 or sse3 should execute on any compatible non-Intel processor with support for the corresponding instruction set.

## **Alternate Options**

Linux and Mac OS X: None

Windows: /arch

#### See Also

x, Qx compiler option

ax, Qax compiler option

arch compiler option

## m32, m64

Tells the compiler to generate code for a specific architecture.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64 architectures

## **Syntax**

Linux and Mac OS X: -m32

-m64

Windows: None

## Arguments

None

### Default

OFF The compiler's behavior depends on the host system.

## Description

These options tell the compiler to generate code for a specific architecture.

Option	Description
-m32	Tells the compiler to generate code for IA-32 architecture.
-m64	Tells the compiler to generate code for Intel® 64 architecture.

The -m32 and -m64 options are the same as Mac OS\* X options -arch i386 and -arch x86\_64, respectively. Note that these options are provided for compatibility with gcc. They are not related to the Intel® Fortran compiler option arch.

## **Alternate Options**

None

#### map

Tells the linker to generate a link map file.

## **IDE Equivalent**

Windows: Linker > Debug > Generate Map File

Linux: None

Mac OS X: None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: None

Windows: /map[:file]

/nomap

## Arguments

file Is the name for the link map file. It can be a file name or a directory name.

#### Default

/nomap No link map is generated.

Description
-------------

This option tells the linker to generate a link map file.

## **Alternate Options**

None

## map-opts, Qmap-opts

Maps one or more compiler options to their equivalent on a different operating system.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux: -map-opts

Mac OS X: None

Windows: /Qmap-opts

## Arguments

None

#### Default

OFF No platform mappings are performed.

## Description

This option maps one or more compiler options to their equivalent on a different operating system. The result is output to stdout.

On Windows systems, the options you provide are presumed to be Windows options, so the options that are output to stdout will be Linux equivalents.

On Linux systems, the options you provide are presumed to be Linux options, so the options that are output to stdout will be Windows equivalents.

The tool can be invoked from the compiler command line or it can be used directly.

No compilation is performed when the option mapping tool is used.

This option is useful if you have both compilers and want to convert scripts or makefiles.



Compiler options are mapped to their equivalent on the architecture you are using.

For example, if you are using a processor with IA-32 architecture, you will only see equivalent options that are available on processors with IA-32 architecture.

#### **Alternate Options**

None

#### Example

The following command line invokes the option mapping tool, which maps the Linux options to Windows-based options, and then outputs the results to stdout:

```
ifort -map-opts -xP -02
```

The following command line invokes the option mapping tool, which maps the Windows options to Linux-based options, and then outputs the results to stdout:

```
ifort /Qmap-opts /QxP /O2
```

#### See Also

Building Applications: Using the Option Mapping Tool

### march

Tells the compiler to generate code for a specified processor.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64 architectures

## Syntax

Linux: -march=processor

Mac OS X: None

Windows: None

### Arguments

processor Is the processor for which the compiler should

generate code. Possible values are:

pentium3 Generates code for Intel®

Pentium® III processors.

pentium4 Generates code for Intel®

Pentium® 4 processors.

core2 Generates code for the Intel®

Core 2<sup>™</sup> processor family.

#### Default

OFF or On IA-32 architecture, the compiler does not

generate processor-specific code unless it is told to

march=pentium4 do so. On systems using Intel® 64 architecture, the

compiler generates code for Intel Pentium 4

processors.

## Description

This option tells the compiler to generate code for a specified processor.

Specifying -march=pentium4 sets -mtune=pentium4.

For compatibility, a number of historical *processor* values are also supported, but the generated code will not differ from the default.

## **Alternate Options**

None

#### mcmodel

Tells the compiler to use a specific memory model to generate code and store data.

## **IDE Equivalent**

None

#### Architectures

Intel® 64 architecture

## Syntax

Linux: -mcmodel=mem\_model

Mac OS X: None Windows: None

#### Arguments

*mem\_model* Is the memory model to use. Possible values are:

small Tells the compiler to restrict

code and data to the first 2GB of address space. All accesses of code and data can be done with Instruction Pointer (IP)-

relative addressing.

medium Tells the compiler to restrict

code to the first 2GB; it places

no memory restriction on data.

Accesses of code can be done

with IP-relative addressing, but

accesses of data must be done

with absolute addressing.

large Places no memory restriction

on code or data. All accesses

of code and data must be done

with absolute addressing.

#### **Default**

On systems using Intel® 64 architecture, the compiler
mcmodel=small restricts code and data to the first 2GB of address
space. Instruction Pointer (IP)-relative addressing can
be used to access code and data.

#### Description

This option tells the compiler to use a specific memory model to generate code and store data. It can affect code size and performance. If your program has COMMON blocks and local data with a total size smaller than 2GB, – mcmodel=small is sufficient. COMMONs larger than 2GB require – mcmodel=medium or -mcmodel=large. Allocation of memory larger than 2GB can be done with any setting of -mcmodel.

IP-relative addressing requires only 32 bits, whereas absolute addressing requires 64-bits. IP-relative addressing is somewhat faster. So, the small memory model has the least impact on performance.



When you specify -mcmodel=medium or -mcmodel=large, you must also specify compiler option -shared-intel to ensure that the correct dynamic versions of the Intel run-time libraries are used.

## **Alternate Options**

None

### Example

The following example shows how to compile using -mcmodel:

```
ifort -shared-intel -mcmodel=medium -o prog prog.f
```

#### See Also

```
shared-intel compiler option
fpic compiler option
```

## mcpu

This is a deprecated option. See mtune.

## MD

Tells the linker to search for unresolved references in a multithreaded, debug, dynamic-link run-time library.

## IDE Equivalent

None

## Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: None

Windows: /MD

/MDd

## Arguments

None

#### Default

OFF The linker searches for unresolved references in a singlethreaded, static run-time library.

## Description

This option tells the linker to search for unresolved references in a multithreaded, debug, dynamic-link (DLL) run-time library. This is the same as specifying options /libs:dll /threads /dbglibs. You can also specify /MDd, where d indicates a debug version.

## **Alternate Options**

None

## See Also

libs compiler option

### MDs

Tells the linker to search for unresolved references in a single-threaded, dynamic-link run-time library.

## **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

#### **Syntax**

Linux and Mac OS X: None

Windows: /MDs

/MDsd

## Arguments

None

#### Default

OFF The linker searches for unresolved references in a singlethreaded, static run-time library.

## Description

This option tells the linker to search for unresolved references in a singlethreaded, dynamic-link (DLL) run-time library.

You can also specify /MDsd, where *d* indicates a debug version.

## **Alternate Options**

/MDs Linux and Mac OS X: None

Windows: /libs:dll

#### See Also

libs compiler option

## mdynamic-no-pic

Generates code that is not position-independent but has position-independent external references.

## **IDE Equivalent**

None

## **Architectures**

## IA-32 architecture

## **Syntax**

Linux: None

Mac OS X: -mdynamic-no-pic

Windows: None

## Arguments

None

#### Default

OFF All references are generated as position independent.

## Description

This option generates code that is not position-independent but has position-independent external references.

The generated code is suitable for building executables, but it is not suitable for building shared libraries.

This option may reduce code size and produce more efficient code. It overrides the -fpic compiler option.

## **Alternate Options**

None

#### See Also

fpic compiler option

### MG

See winapp.

## mieee-fp

See <u>fltconsistency</u>.

## minstruction, Qinstruction

Determines whether MOVBE instructions are generated for Intel processors.

### **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64 architectures

## Syntax

Linux and Mac OS X: -minstruction=[no]movbe

Windows: /Qinstruction:[no]movbe

### Arguments

None

#### Default

-minstruction=movbe The compiler generates MOVBE or/Qinstruction:movbe instructions for Intel® Atom™ processors.

#### Description

This option determines whether MOVBE instructions are generated for Intel processors. To use this option, you must also specify -xSSE3\_ATOM (Linux and Mac OS X) or /QxSSE3\_ATOM (Windows).

If -minstruction=movbe or /Qinstruction:movbe is specified, the following occurs:

MOVBE instructions are generated that are specific to the Intel® Atom™
processor.

- The options are ON by default when -xSSE3\_ATOM or /QxSSE3\_ATOM is specified.
- Generated executables can only be run on Intel® Atom™ processors or processors that support Intel® Streaming SIMD Extensions 3 (Intel® SSE3) and MOVBE.

If -minstruction=nomovbe or /Qinstruction:nomovbe is specified, the following occurs:

- The compiler optimizes code for the Intel® Atom™ processor, but it does not generate MOVBE instructions.
- Generated executables can be run on non-Intel® Atom™ processors that support Intel® SSE3.

## **Alternate Options**

None

#### See Also

x, Qx compiler option

## mixed-str-len-arg

See <u>iface</u>.

#### ML

Tells the linker to search for unresolved references in a single-threaded, static run-time library.

This option has been deprecated.

## IDE Equivalent

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: None

Windows: /ML

/MLd

#### Arguments

None

#### Default

Systems using Intel® On systems using Intel® 64 architecture, the linker

64 architecture: OFF searches for unresolved references in a

Systems using IA-32 multithreaded, static run-time library. On systems

architecture and IA- using IA-32 architecture and IA-64 architectures,

64 architecture: /ML. the linker searches for unresolved references in a

single-threaded, static run-time library.

## Description

This option tells the linker to search for unresolved references in a single-threaded, static run-time library. You can also specify /MLd, where d indicates a debug version.

#### **Alternate Options**

Linux: None

Mac OS X: None

Windows: /libs:static

#### See Also

libs compiler option

### module

Specifies the directory where module files should be placed when created and where they should be searched for.

## **IDE Equivalent**

Windows: Output > Module Path

Linux: None

Mac OS X: Output Files > Module Path

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -module path

Windows: /module:path

## Arguments

path Is the directory for module files.

#### Default

OFF The compiler places module files in the current directory.

#### Description

This option specifies the directory (path) where module (.mod) files should be placed when created and where they should be searched for (USE statement).

## **Alternate Options**

None

## mp

See <u>fltconsistency</u>

## multiple-processes, MP

Creates multiple processes that can be used to compile large numbers of source files at the same time.

ID	EΕ	aui	val	ent

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -multiple-processes[=n]

Windows: /MP[:n]

#### Arguments

*n* Is the maximum number of processes that the

compiler should create.

#### Default

OFF A single process is used to compile source files.

#### Description

This option creates multiple processes that can be used to compile large numbers of source files at the same time. It can improve performance by reducing the time it takes to compile source files on the command line.

This option causes the compiler to create one or more copies of itself, each in a separate process. These copies simultaneously compile the source files.

If n is not specified for this option, the default value is as follows:

- On Windows OS, the value is based on the setting of the NUMBER OF PROCESSORS environment variable.
- On Linux OS and Mac OS X, the value is 2.

This option applies to compilations, but not to linking or link-time code generation.

## **Alternate Options**

None

## mp1, Qprec

Improves floating-point precision and consistency.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

#### **Syntax**

Linux and Mac OS X: -mp1

Windows: /Qprec

### Arguments

None

#### Default

OFF The compiler provides good accuracy and run-time performance at the expense of less consistent floating-point results.

## Description

This option improves floating-point consistency. It ensures the out-of-range check of operands of transcendental functions and improves the accuracy of floating-point compares.

This option prevents the compiler from performing optimizations that change NaN comparison semantics and causes all values to be truncated to declared precision before they are used in comparisons. It also causes the compiler to use library routines that give better precision results compared to the X87 transcendental instructions.

This option disables fewer optimizations and has less impact on performance than option fltconsistency or mp.

Alternate Options		
None		
See Also		
fltconsistency compiler option  mp compiler option		
mrelax		
Tells the compiler to	pass linker option -relax to the linker.	
IDE Equivalent		
None		
Architectures		
IA-64 architecture		
Syntax		
Linux and Mac OS X	(:-mrelax	
	-mno-relax	
Mac OS X:	None	
Windows:	None	
Arguments		
None		
Default		
-mno-relax The compiler does not pass -relax to the linker.		
Description		
This option tells the compiler to pass linker option -relax to the linker.		
Alternate Options		

Ν	0	n	е
---	---	---	---

### MT

Tells the linker to search for unresolved references in a multithreaded, static runtime library.

#### **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: None

Windows: /MT

/MTd

# **Arguments**

None

#### Default

Systems using Intel® On systems using Intel® 64 architecture, the linker

64 architecture: /MT searches for unresolved references in a

/noreentrancy multithreaded, static run-time library. On systems

IA-32 architecture using IA-32 architecture and IA-64 architecture,

and IA-64 the linker searches for unresolved references in a

architecture: OFF single-threaded, static run-time library. However,

on systems using IA-32 architecture, if option

Qvc8 is in effect, the linker searches for

unresolved references in threaded libraries.

#### Description

This option tells the linker to search for unresolved references in a multithreaded, static run-time library. This is the same as specifying options /libs:static /threads /noreentrancy. You can also specify /MTd, where d indicates a debug version.

#### **Alternate Options**

None

#### See Also

Ovc compiler option
libs compiler option
threads compiler option
reentrancy compiler option

#### mtune

Performs optimizations for specific processors.

# **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -mtune=processor

Windows: None

# Arguments

processor Is the processor for which the compiler should

perform optimizations. Possible values are:

generic Generates code for the

compiler's default behavior.

core2 Optimizes for the Intel® Core 2

processor family, including support for MMX<sup>™</sup>, Intel®

SSE, SSE2, SSE3 and SSSE3

instruction sets.

pentium Optimizes for Intel® Pentium®

processors.

pentium- Optimizes for Intel® Pentium®

mmx with MMX technology.

pentiumpro Optimizes for Intel® Pentium®

Pro, Intel Pentium II, and Intel

Pentium III processors.

pentium4 Optimizes for Intel® Pentium®

4 processors.

pentium4m Optimizes for Intel® Pentium®

4 processors with MMX

technology.

itanium2 Optimizes for Intel® Itanium®

2 processors.

itanium2- Optimizes for the Dual-Core

p9000 Intel® Itanium® 2 processor

9000 series. This option

affects the order of the

generated instructions, but the

generated instructions are

limited to Intel® Itanium® 2

processor instructions unless

the program uses (executes)

intrinsics specific to the Dual-Core Intel® Itanium® 2 processor 9000 series.

#### Default

generic On systems using IA-32 and Intel® 64 architectures, code is generated for the compiler's default behavior.
 itanium2- On systems using IA-64 architecture, the compiler optimizes for the Dual-Core Intel® Itanium® 2 processor 9000 series.

# Description

This option performs optimizations for specific processors.

The resulting executable is backwards compatible and generated code is optimized for specific processors. For example, code generated with - mtune=itanium2-p9000 will run correctly on single-core Itanium® 2 processors, but it might not run as fast as if it had been generated using - mtune=itanium2.

The following table shows on which architecture you can use each value.

	Architecture	ecture			
processor Value	IA-32 architecture	Intel® 64 architecture	IA-64 architecture		
generic	X	X	X		
core2	X	Χ			
pentium	Χ				
pentium-mmx	X				
pentiumpro	X				
pentium4	X				

	Architecture		
processor Value	IA-32 architecture	Intel® 64 architecture	IA-64 architecture
pentium4m	Х		
itanium2			X
itanium2-			X
p9000			

### **Alternate Options**

-mtune Linux: -mcpu (this is a deprecated option)

Mac OS X: None Windows: None

-mtune=itanium2 Linux: -mcpu=itanium2 (-mcpu is a deprecated

option)

Mac OS X: None Windows: /G2

- Linux: -mcpu=itanium2-p9000 (-mcpu is a

deprecated option)
Mac OS X: None

p9000 Windows: /G2-p9000

# multiple-processes, MP

mtune=itanium2-

Creates multiple processes that can be used to compile large numbers of source files at the same time.

# **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -multiple-processes[=n]

Windows: /MP[:n]

### Arguments

*n* Is the maximum number of processes that the

compiler should create.

#### Default

OFF A single process is used to compile source files.

# Description

This option creates multiple processes that can be used to compile large numbers of source files at the same time. It can improve performance by reducing the time it takes to compile source files on the command line.

This option causes the compiler to create one or more copies of itself, each in a separate process. These copies simultaneously compile the source files.

If n is not specified for this option, the default value is as follows:

- On Windows OS, the value is based on the setting of the NUMBER\_OF\_PROCESSORS environment variable.
- On Linux OS and Mac OS X, the value is 2.

This option applies to compilations, but not to linking or link-time code generation.

#### **Alternate Options**

None

#### MW

See <u>libs</u>.

# MWs

See libs.

#### names

Specifies how source code identifiers and external names are interpreted.

# **IDE Equivalent**

Windows: External Procedures > Name Case Interpretation

Linux: None

Mac OS X: External Procedures > Name Case Interpretation

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -names keyword

Windows: /names:keyword

#### **Arguments**

keyword Specifies how to interpret the identifiers and external names in source code. Possible values are:

lowercase Causes the compiler to ignore case

differences in identifiers and to convert

external names to lowercase.

uppercase Causes the compiler to ignore case

differences in identifiers and to convert

external names to uppercase.

as\_is Causes the compiler to distinguish

case differences in identifiers and to

preserve the case of external names.

# Default

lowercase This is the default on Linux and Mac OS X

systems. The compiler ignores case differences in

identifiers and converts external names to

lowercase.

uppercase This is the default on Windows systems. The

compiler ignores case differences in identifiers and

converts external names to uppercase.

# Description

This option specifies how source code identifiers and external names are interpreted. It can be useful in mixed-language programming.

This naming convention applies whether names are being defined or referenced. You can use the ALIAS directive to specify an alternate external name to be used when referring to external subprograms.



On Windows systems, if you specify option /iface:cref, it overrides the default for external names and causes them to be lowercase. It is as if you specified "!dec\$ attributes c, reference" for the external name. If you specify option /iface:cref and want external names to be uppercase, you must explicitly specify option /names:uppercase.

#### **Alternate Options**

names lowercase Linux and Mac OS X: -lowercase

Windows: /Qlowercase

names uppercase Linux and Mac OS X: -uppercase

Windows: /Quppercase

#### See Also

iface compiler option

# **ALIAS Directive**

#### nbs

See assume.

# no-bss-init, Qnobss-init

Tells the compiler to place in the DATA section any variables explicitly initialized with zeros.

# **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -no-bss-init

Windows: /Qnobss-init

# Arguments

None

# Default

OFF Variables explicitly initialized with zeros are placed in the BSS section.

# Description

This option tells the compiler to place in the DATA section any variables explicitly initialized with zeros.

# **Alternate Options**

Linux and Mac OS X: -nobss-init (this is a deprecated option)

Intel® Fortran Compiler User and Reference Guides
Windows: None
nodefaultlibs
Prevents the compiler from using standard libraries when linking.
IDE Equivalent
None
Architectures
IA-32, Intel® 64, IA-64 architectures
Syntax
Linux and Mac OS X: -nodefaultlibs
Windows: None
Arguments
None
Default
OFF The standard libraries are linked.
Description
This option prevents the compiler from using standard libraries when linking.
Alternate Options
None
See Also
nostdlib compiler option
nodefine
See <u>D</u> .

#### nofor-main

Specifies that the main program is not written in Fortran.

# **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -nofor-main

Windows: None

# Arguments

None

# Default

OFF The compiler assumes the main program is written in Fortran.

# Description

This option specifies that the main program is not written in Fortran. It is a link-time option that prevents the compiler from linking  $for_{main.o}$  into applications. For example, if the main program is written in C and calls a Fortran subprogram, specify  $-nofor_{main}$  when compiling the program with the ifort command. If you omit this option, the main program must be a Fortran program.

# **Alternate Options**

None

# noinclude

See X.

no	п	h-I	ni	III	

Disables inline expansion of standard library or intrinsic functions.

# **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -nolib-inline

Windows: None

# Arguments

None

#### Default

OFF The compiler inlines many standard library and intrinsic functions.

# Description

This option disables inline expansion of standard library or intrinsic functions. It prevents the unexpected results that can arise from inline expansion of these functions.

# **Alternate Options**

None

#### nostartfiles

Prevents the compiler from using standard startup files when linking.

# **IDE Equivalent**

None

330

Architectures
IA-32, Intel® 64, IA-64 architectures
Syntax
Linux and Mac OS X: -nostartfiles
Windows: None
Arguments
None
Default
OFF The compiler uses standard startup files when linking.
Description
This option prevents the compiler from using standard startup files when linking.
Alternate Options
None
See Also
nostdlib compiler option
nostdinc
See <u>X</u> .
nostdlib
Prevents the compiler from using standard libraries and startup files when linking.
IDE Equivalent
None
Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -nostdlib

Windows: None

# Arguments

None

#### Default

OFF The compiler uses standard startup files and standard libraries when linking.

# Description

This option prevents the compiler from using standard libraries and startup files when linking.

# **Alternate Options**

None

# See Also

nodefaultlibs compiler option
nostartfiles compiler option

#### nus

See <u>assume</u>.

0

Specifies the name for an output file.

# IDE Equivalent

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -ofile

Windows: None

#### Arguments

file Is the name for the output file. The space before

*file* is optional.

#### Default

OFF The compiler uses the default file name for an output file.

#### Description

This option specifies the name for an output file as follows:

- If -c is specified, it specifies the name of the generated object file.
- If -S is specified, it specifies the name of the generated assembly listing file.
- If -preprocess-only or -P is specified, it specifies the name of the generated preprocessor file.

Otherwise, it specifies the name of the executable file.



If you misspell a compiler option beginning with "o", such as <code>-openmp</code>, <code>-opt-report</code>, etc., the compiler interprets the misspelled option as an <code>-ofile</code> option. For example, say you misspell "<code>-opt-report</code>" as "<code>-opt-reprt</code>"; in this case, the compiler interprets the misspelled option as "<code>-off-reprt</code>", where pt-reprt is the output file name.

# **Alternate Options**

Linux and Mac OS X: None

Windows: /Fe, /exe

#### See Also

<u>Fe</u> compiler option

object compiler option

0

Specifies the code optimization for applications.

# **IDE Equivalent**

Windows: **General > Optimization** (/Od, /O1, /O2, /O3, /fast)

Optimization > Optimization (/od, /o1, /o2, /o3, /fast)

Linux: None

Mac OS X: **General > Optimization Level** (−○)

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -0[n]

Windows: /O[n]

#### Arguments

*n* Is the optimization level. Possible values are 1, 2,

or 3. On Linux and Mac OS X systems, you can

also specify 0.

# Default

Optimizes for code speed. This default may change depending on which other compiler options are specified. For details, see

below.

# Description

This option specifies the code optimization for applications.

Option	Description
o (Linux and Mac OS X)	This is the same as specifying O2.
O0 (Linux and Mac OS X)	Disables all optimizations. On systems using IA-32 architecture and Intel® 64 architecture, this option sets option -fno-omit-frame-pointer and option -fmath-errno.  Option -00 also implies option -mp. So, intermediate floating-point results are evaluated at extended precision. On IA-32 or Intel® 64 architecture, this may cause the compiler to use x87 instructions instead of SSE instructions. You can use option -fp-model to independently control the evaluation precision for intermediate results.  Option -mp is deprecated. Therefore, the default behavior of -00 may change in a future compiler release.  This option causes certain warn options to be ignored. This is the default if you specify option -debug (with no keyword).
01	<ul> <li>Enables optimizations for speed and disables some optimizations that increase code size and affect speed.</li> <li>To limit code size, this option:</li> <li>Enables global optimization; this includes data-flow analysis, code motion, strength reduction and test</li> </ul>

# Option Description replacement, split-lifetime analysis, and instruction scheduling. On systems using IA-64 architecture, it disables software pipelining, loop unrolling, and global code scheduling. On systems using IA-64 architecture, this option also enables optimizations for server applications (straightline and branch-like code with a flat profile). The 01 option sets the following options: On Linux and Mac OS X systems: -funroll-loops0, -nofltconsistency (same as -mno-ieee-fp), -fomit-frame-pointer, ftz • On Windows systems using IA-32 architecture: /Qunrollo, /nofltconsistency (same as /Op-), /Oy, /Os, /Ob2, /Qftz On Windows systems using Intel® 64 architecture and IA-64 architecture: /Qunrollo, /nofltconsistency (same as /Op-), /Os, /Ob2, /Qftz The O1 option may improve performance for applications with very large code size, many branches, and execution time not dominated by code within loops. 02

Enables optimizations for speed. This is the generally recommended optimization level.

Vectorization is enabled at O2 and higher levels.

On systems using IA-64 architecture, this option enables optimizations for speed, including global code scheduling, software pipelining, predication, and

Option	Description
	speculation.
	This option also enables:
	Inlining of intrinsics
	• Intra-file interprocedural optimization, which includes:
	o inlining
	o constant propagation
	<ul> <li>forward substitution</li> </ul>
	<ul> <li>routine attribute propagation</li> </ul>
	<ul> <li>variable address-taken analysis</li> </ul>
	<ul> <li>dead static function elimination</li> </ul>
	<ul> <li>removal of unreferenced variables</li> </ul>
	The following capabilities for performance gain:
	o constant propagation
	o copy propagation
	o dead-code elimination
	<ul> <li>global register allocation</li> </ul>
	<ul> <li>global instruction scheduling and control speculation</li> </ul>
	o loop unrolling
	o optimized code selection
	<ul> <li>partial redundancy elimination</li> </ul>
	<ul> <li>strength reduction/induction variable simplification</li> </ul>
	<ul> <li>variable renaming</li> </ul>
	<ul> <li>exception handling optimizations</li> </ul>
	o tail recursions
	<ul> <li>peephole optimizations</li> </ul>
	<ul> <li>structure assignment lowering and optimizations</li> </ul>
	<ul> <li>dead store elimination</li> </ul>
	On Windows systems, this option is the same as the $\mathtt{Ox}$
	option.

Option	Description
	The 02 option sets the following options:
	<ul> <li>On Windows systems using IA-32 architecture:</li> </ul>
	/Og, /Ot, /Oy, /Ob2, /Gs, and /Qftz
	<ul> <li>On Windows systems using Intel® 64 architecture:</li> </ul>
	/Og, /Ot, /Ob2, /Gs, and /Qftz
	On Linux and Mac OS X systems, if -g is specified, 02 is
	turned off and 00 is the default unless 02 (or 01 or 03) is
	explicitly specified in the command line together with -g.
	This option sets other options that optimize for code
	speed. The options set are determined by the compiler
	depending on which architecture and operating system
	you are using.
03	Enables 02 optimizations plus more aggressive
	optimizations, such as prefetching, scalar replacement,
	and loop and memory access transformations. Enables
	optimizations for maximum speed, such as:
	Loop unrolling, including instruction scheduling
	Code replication to eliminate branches
	Padding the size of certain power-of-two arrays to
	allow more efficient cache use.
	On Windows systems, the O3 option sets the /Ob2
	option.
	On Linux and Mac OS X systems, the O3 option sets
	option -fomit-frame-pointer.
	On systems using IA-32 architecture or Intel® 64
	architecture, when o3 is used with options -ax or -x
	(Linux) or with options /Qax or /Qx (Windows), the
	compiler performs more aggressive data dependency

Option	Description
	analysis than for O2, which may result in longer
	compilation times.
	On systems using IA-64 architecture, the o3 option
	enables optimizations for technical computing
	applications (loop-intensive code): loop optimizations
	and data prefetch.
	The O3 optimizations may not cause higher performance
	unless loop and memory access transformations take
	place. The optimizations may slow down code in some
	cases compared to 02 optimizations.
	The O3 option is recommended for applications that have
	loops that heavily use floating-point calculations and
	process large data sets.

The last o option specified on the command line takes precedence over any others.



The options set by the o option may change from release to release.

# **Alternate Options**

01	Linux and Mac OS X: None
	Windows: /Od, /optimize:0, /nooptimize
02	Linux and Mac OS X: None
	Windows: /optimize:1, /optimize:2
03	Linux and Mac OS X: None
	Windows: /Ox, /optimize: 3, /optimize: 4
04	Linux and Mac OS X: None

Windows: /optimize:5

#### See Also

od compiler option

Op compiler option

fp-model, fp compiler option

fast compiler option

#### See Also

Optimizing Applications:

Compiler Optimizations Overview

**Optimization Options Summary** 

Efficient Compilation

# inline-level, Ob

Specifies the level of inline function expansion.

# **IDE Equivalent**

Windows: **Optimization > Inline Function Expansion** 

Linux: None

Mac OS X: Optimization > Inline Function Expansion

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -inline-level=n

Windows: /Obn

#### Arguments

*n* Is the inline function expansion level. Possible

values are 0, 1, and 2.

#### Default

-inline- This is the default if option O2 is specified or is in effect by level=2 or default. On Windows systems, this is also the default if option O3 is specified.

-inline- This is the default if option -O0 (Linux and Mac OS) or level=0 or /Od (Windows) is specified.

/Ob0

# Description

This option specifies the level of inline function expansion. Inlining procedures can greatly improve the run-time performance of certain programs.

Option	Description
-inline-level=0	Disables inlining of user-defined functions. Note that
or Ob0	statement functions are always inlined.
-inline-level=1	Enables inlining when an inline keyword or an inline
or Obl	directive is specified.
-inline-level=2	Enables inlining of any function at the compiler's
or Ob2	discretion.

#### **Alternate Options**

Linux: -Ob (this is a deprecated option)

Mac OS X: None Windows: None

#### See Also

inline compiler option

# object

Specifies the name for an object file.

# **IDE Equivalent**

Windows: Output Files > Object File Name

Linux: None

Mac OS X: None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: None

Windows: /object:file

#### Arguments

file Is the name for the object file. It can be a file or directory name.

#### Default

OFF An object file has the same name as the name of the first source file and a file extension of .obj.

# Description

This option specifies the name for an object file.

If you specify this option and you omit /c or /compile-only, the /object option gives the object file its name.

On Linux and Mac OS X systems, this option is equivalent to specifying option – ofile -c.

#### **Alternate Options**

Linux and Mac OS X: None

Windows: /Fo

#### See Also

o compiler option

#### Od

Disables all optimizations.

#### **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

#### **Syntax**

Linux and Mac OS X: None

Windows: /Od

### **Arguments**

None

#### **Default**

OFF The compiler performs default optimizations.

### Description

This option disables all optimizations. It can be used for selective optimizations, such as a combination of /Od and /Og (disables all optimizations except global optimizations), or /Od and /Ob1 (disables all optimizations, but enables inlining). This option also causes certain /warn options to be ignored.

On IA-32 architecture, this option sets the /Oy- option.

Option /od also implies option /op. So, intermediate floating-point results are evaluated at extended precision. On IA-32 and Intel® 64 architecture, this may cause the compiler to use x87 instructions instead of SSE instructions. You can

use option /fp to independently control the evaluation precision for intermediate

results.

Option / Op is deprecated. Therefore, the default behavior of / Od may change in

a future compiler release.

# **Alternate Options**

Linux and Mac OS X: -00

Windows: /optimize:0

#### See Also

o compiler option (see O0)

fp-model, fp compiler option

# Og

Enables global optimizations.

# **IDE Equivalent**

Windows: Optimization > Global Optimizations

Linux: None

Mac OS X: None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: None

Windows: /0g

/0g-

# Arguments

None

# Default

/Og Global optimizations are enabled unless /Od is specified.

# Description

This option enables global optimizations.

# **Alternate Options**

None

# onetrip, Qonetrip

Tells the compiler to follow the FORTRAN 66 Standard and execute DO loops at least once.

# **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -onetrip

Windows: /Qonetrip

# **Arguments**

None

#### Default

OFF The compiler applies the current Fortran Standard semantics, which allows zero-trip DO loops.

# Description

This option tells the compiler to follow the FORTRAN 66 Standard and execute DO loops at least once.

# **Alternate Options**

Linux and Mac OS X: -1

Windows: /1

# Op

This is a deprecated option. See <u>fltconsistency</u>.

# openmp, Qopenmp

Enables the parallelizer to generate multi-threaded code based on the OpenMP\* directives.

# **IDE Equivalent**

Windows: Language > Process OpenMP Directives

Linux: None

Mac OS X: None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -openmp

Windows: /Qopenmp

# Arguments

None

#### Default

OFF No OpenMP multi-threaded code is generated by the compiler.

# Description

This option enables the parallelizer to generate multi-threaded code based on the OpenMP\* directives. The code can be executed in parallel on both uniprocessor and multiprocessor systems.

If you use this option, multithreaded libraries are used, but option fpp is not automatically invoked.

This option sets option automatic.

This option works with any optimization level. Specifying no optimization (-00 on Linux or /od on Windows) helps to debug OpenMP applications.



On Mac OS X systems, when you enable OpenMP\*, you must also set the DYLD\_LIBRARY\_PATH environment variable within Xcode or an error will be displayed.

# **Alternate Options**

None

#### See Also

openmp-stubs, Qopenmp-stubs compiler option

# openmp-lib, Qopenmp-lib

Lets you specify an OpenMP\* run-time library to use for linking.

# **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux: -openmp-lib type

Mac OS X: None

Windows: /Qopenmp-lib:type

### Arguments

*type* Specifies the type of library to use; it implies

compatibility levels. Possible values are:

legacy Tells the compiler to use the

legacy OpenMP\* run-time library (libguide). This setting does not provide compatibility with object files created using

other compilers. This is a

<u>deprecated</u> option.

compat Tells the compiler to use the

compatibility OpenMP\* runtime library (libiomp). This setting provides compatibility with object files created using

Microsoft\* and GNU\*

compilers.

#### Default

-openmp-lib compat The compiler uses the compatibility or/Qopenmp-lib:compat OpenMP\* run-time library (libiomp).

#### Description

This option lets you specify an OpenMP\* run-time library to use for linking.

The legacy OpenMP run-time library is not compatible with object files created using OpenMP run-time libraries supported in other compilers.

The compatibility OpenMP run-time library is compatible with object files created using the Microsoft\* OpenMP run-time library (vcomp) and GNU OpenMP run-time library (libgomp).

To use the compatibility OpenMP run-time library, compile and link your application using the <code>-openmp-lib</code> compat (Linux) or <code>/Qopenmp-lib:compat</code> (Windows) option. To use this option, you must also specify one of the following compiler options:

- Linux OS: -openmp, -openmp-profile, or -openmp-stubs
- Windows OS: /Qopenmp, /Qopenmp-profile, or /Qopenmp-stubs
  On Windows\* systems, the compatibility OpenMP\* run-time library lets you
  combine OpenMP\* object files compiled with the Microsoft\* C/C++ compiler with
  OpenMP\* object files compiled with the Intel C/C++ or Fortran compilers. The
  linking phase results in a single, coherent copy of the run-time library.
  On Linux\* systems, the compatibility Intel OpenMP\* run-time library lets you
  combine OpenMP\* object files compiled with the GNU\* gcc or gfortran compilers
  with similar OpenMP\* object files compiled with the Intel C/C++ or Fortran
  compilers. The linking phase results in a single, coherent copy of the run-time
  library.

You cannot link object files generated by the Intel® Fortran compiler to object files compiled by the GNU Fortran compiler, regardless of the presence or absence of the <code>-openmp</code> (Linux) or <code>/Qopenmp</code> (Windows) compiler option. This is because the Fortran run-time libraries are incompatible.



The compatibility OpenMP run-time library is not compatible with object files created using versions of the Intel compiler earlier than 10.0.

#### **Alternate Options**

None

#### See Also

```
openmp, Qopenmp compiler option
openmp-stubs, Qopenmp-stubs compiler option
openmp-profile, Qopenmp-profile compiler option
```

# openmp-link, Qopenmp-link

Controls whether the compiler links to static or dynamic OpenMP run-time libraries.

# **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -openmp-link library

Windows: /Qopenmp-link:library

# Arguments

library Specifies the OpenMP library to use. Possible

values are:

static Tells the compiler to link to

static OpenMP run-time

libraries.

dynamic Tells the compiler to link to

dynamic OpenMP run-time

libraries.

#### Default

-openmp-link dynamic The compiler links to dynamic OpenMP or /Qopenmp- run-time libraries. However, if option

link:dynamic

static is specified, the compiler links to

static OpenMP run-time libraries.

### Description

This option controls whether the compiler links to static or dynamic OpenMP runtime libraries.

To link to the static OpenMP run-time library (RTL) and create a purely static executable, you must specify <code>-openmp-link</code> static (Linux and Mac OS X) or <code>/Qopenmp-link:static</code> (Windows). However, we strongly recommend you use the default setting, <code>-openmp-link</code> dynamic (Linux and Mac OS X) or <code>/Qopenmp-link:dynamic</code> (Windows).



Compiler options -static-intel and -shared-intel (Linux and Mac OS X) have no effect on which OpenMP run-time library is linked.

#### **Alternate Options**

None

# openmp-profile, Qopenmp-profile

Enables analysis of OpenMP\* applications if Intel® Thread Profiler is installed.

# IDE Equivalent

None

# Architectures

IA-32, Intel® 64, IA-64 architectures

#### Syntax

Linux: -openmp-profile

Mac OS X: None

Windows:

/Qopenmp-profile

Arguments

None

Default

OFF OpenMP applications are not analyzed.

Description

This option enables analysis of OpenMP\* applications. To use this option, you must have previously installed Intel® Thread Profiler, which is one of the Intel® Threading Analysis Tools.

This option can adversely affect performance because of the additional profiling and error checking invoked to enable compatibility with the threading tools. Do not use this option unless you plan to use the Intel® Thread Profiler.

For more information about Intel® Thread Profiler (including an evaluation copy) open the page associated with threading tools at <a href="Intel® Software Development">Intel® Software Development</a> Products.

**Alternate Options** 

None

openmp-report, Qopenmp-report

Controls the OpenMP\* parallelizer's level of diagnostic messages.

**IDE Equivalent** 

Windows: Compilation Diagnostics > OpenMP Diagnostic Level

Linux: None

Mac OS X: Compiler Diagnostics > OpenMP Report

Architectures

# IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -openmp-report[n]

Windows: /Qopenmp-report[n]

# Arguments

*n* Is the level of diagnostic messages to display.

Possible values are:

0 No diagnostic messages are

displayed.

1 Diagnostic messages are

displayed indicating loops,

regions, and sections

successfully parallelized.

2 The same diagnostic

messages are displayed as

specified by openmp\_report1

plus diagnostic messages

indicating successful handling

of MASTER constructs,

SINGLE constructs, CRITICAL

constructs, ORDERED

constructs, ATOMIC directives,

and so forth.

#### Default

-openmp-report 1 This is the default if you do not specify n.

or/Qopenmp-report1 The compiler displays diagnostic messages

indicating loops, regions, and sections successfully parallelized. If you do not specify the option on the command line, the default is to display no messages.

# Description

This option controls the OpenMP\* parallelizer's level of diagnostic messages. To use this option, you must also specify <code>-openmp</code> (Linux and Mac OS X) or <code>/Qopenmp</code> (Windows).

If this option is specified on the command line, the report is sent to stdout.

On Windows systems, if this option is specified from within the IDE, the report is included in the build log if the Generate Build Logs option is selected.

# **Alternate Options**

None

### See Also

<u>openmp</u>, <u>Qopenmp</u> compiler option Optimizing Applications:

Using Parallelism

OpenMP\* Report

# openmp-stubs, Qopenmp-stubs

Enables compilation of OpenMP programs in sequential mode.

# **IDE Equivalent**

Windows: Language > Process OpenMP Directives

Linux: None

Mac OS X: Language > Process OpenMP Directives

### **Architectures**

IA-32, Intel® 64, IA-64 architectures		
Syntax		
Linux and Mac OS X: -openmp-stubs		
Windows: /Qopenmp-stubs		
Arguments		
None		
Default		
OFF The library of OpenMP function stubs is not linked.		
Description		
This option enables compilation of OpenMP programs in sequential mode. The OpenMP directives are ignored and a stub OpenMP library is linked.		
Alternate Options		
None		
See Also		
openmp, Qopenmp compiler option		
openmp-threadprivate, Qopenmp-threadprivate		
Lets you specify an OpenMP* threadprivate implementation.		
IDE Equivalent		
None		
Architectures		
IA-32, Intel® 64, IA-64 architectures		
Syntax		

Linux: -openmp-threadprivate type

Mac OS X: None

Windows: /Qopenmp-threadprivate:type

#### Arguments

*type* Specifies the type of threadprivate implementation.

Possible values are:

legacy Tells the compiler to use the

legacy OpenMP\* threadprivate

implementation used in the

previous releases of the Intel®

compiler. This setting does not provide compatibility with the

implementation used by other

compilers.

compat Tells the compiler to use the

compatibility OpenMP\*

threadprivate implementation

based on applying the thread-

local attribute to each

threadprivate variable. This

setting provides compatibility

with the implementation

provided by the Microsoft\* and

GNU\* compilers.

#### Default

-openmp-threadprivate The compiler uses the legacy OpenMP\*
legacy threadprivate implementation used in the
or/Qopenmp- previous releases of the Intel® compiler.

threadprivate:legacy

#### Description

This option lets you specify an OpenMP\* threadprivate implementation.

The legacy OpenMP run-time library is not compatible with object files created using OpenMP run-time libraries supported in other compilers.

To use this option, you must also specify one of the following compiler options:

- Linux OS: -openmp, -openmp-profile, or -openmp-stubs
- Windows OS: /Qopenmp, /Qopenmp-profile, or /Qopenmp-stubs

The value specified for this option is independent of the value used for option – openmp-lib (Linux) or /Qopenmp-lib (Windows).

## **Alternate Options**

None

## opt-block-factor, Qopt-block-factor

Lets you specify a loop blocking factor.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

#### **Syntax**

Linux and Mac OS X: -opt-block-factor=n

Windows: /Qopt-block-factor:n

#### Arguments

*n* Is the blocking factor. It must be an integer. The

compiler may ignore the blocking factor if the value

is 0 or 1.

#### Default

OFF The compiler uses default heuristics for loop blocking.

## Description

This option lets you specify a loop blocking factor.

## **Alternate Options**

None

# opt-jump-tables, Qopt-jump-tables

Enables or disables generation of jump tables for switch statements.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -opt-jump-tables=keyword

-no-opt-jump-tables

Windows: /Qopt-jump-tables:keyword

/Qopt-jump-tables-

## Arguments

*keyword* Is the instruction for generating jump tables.

Possible values are:

never Tells the compiler to never

generate jump tables. All

switch statements are implemented as chains of if-then-elses. This is the same as specifying -no-opt-jump-tables (Linux and Mac OS) or /Qopt-jump-tables- (Windows).

default The compiler uses default

heuristics to determine when to

generate jump tables.

large Tells the compiler to generate

jump tables up to a certain pre-

defined size (64K entries).

n Must be an integer. Tells the

compiler to generate jump

tables up ton entries in size.

#### Default

-opt-jump-tables=default The compiler uses default heuristics or/Qopt-jump-tables:default to determine when to generate jump tables for switch statements.

## Description

This option enables or disables generation of jump tables for switch statements. When the option is enabled, it may improve performance for programs with large switch statements.

#### **Alternate Options**

None

# opt-loadpair, Qopt-loadpair

Enables or disables loadpair optimization.

## **IDE Equivalent**

None

#### Architectures

IA-64 architecture

## Syntax

Linux: -opt-loadpair

-no-opt-loadpair

Mac OS X: None

Windows: /Qopt-loadpair

/Qopt-loadpair-

## Arguments

None

#### Default

-no-opt-loadpair Loadpair optimization is disabled

or/Qopt-loadpair- unless option 03 is specified.

## Description

This option enables or disables loadpair optimization.

When -03 is specified on IA-64 architecture, loadpair optimization is enabled by default. To disable loadpair generation, specify -no-opt-loadpair (Linux) or /Qopt-loadpair- (Windows).

## **Alternate Options**

None

# opt-malloc-options

Lets you specify an alternate algorithm for malloc().

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -opt-malloc-options=n

Windows: None

## Arguments

*n* Specifies the algorithm to use for malloc().

Possible values are:

- Tells the compiler to use the default algorithm for malloc(). This is the default.
- Causes the following adjustments to the malloc() algorithm:
   M\_MMAP\_MAX=2 and
   M\_TRIM\_THRESHOLD=0x10000000.
- Causes the following adjustments to the malloc() algorithm: M\_MMAP\_MAX=2 and M\_TRIM\_THRESHOLD=0x40000000.
- Causes the following adjustments to the malloc() algorithm:
  M\_MMAP\_MAX=0 and

M\_TRIM\_THRESHOLD=-1.

4 Causes the following adjustments to the malloc() algorithm: M\_MMAP\_MAX=0, M\_TRIM\_THRESHOLD=-1, M TOP PAD=4096.

#### Default

-opt-malloc-options=0 The compiler uses the default algorithm when malloc() is called. No call is made to mallopt().

## Description

This option lets you specify an alternate algorithm for malloc(). If you specify a non-zero value for n, it causes alternate configuration parameters to be set for how malloc() allocates and frees memory. It tells the compiler to

insert calls to mallopt() to adjust these parameters to malloc() for dynamic

memory allocation. This may improve speed.

## **Alternate Options**

None

#### See Also

malloc(3) man page
mallopt function (defined in malloc.h)

# opt-mem-bandwidth, Qopt-mem-bandwidth

Enables performance tuning and heuristics that control memory bandwidth use among processors.

#### **IDE Equivalent**

None

#### **Architectures**

#### IA-64 architecture

## Syntax

Linux: -opt-mem-bandwidthn

Mac OS X: None

Windows: /Qopt-mem-bandwidthn

#### Arguments

*n* Is the level of optimizing for memory bandwidth

usage. Possible values are:

0 Enables a set of performance

tuning and heuristics in

compiler optimizations that is

optimal for serial code.

1 Enables a set of performance

tuning and heuristics in

compiler optimizations for

multithreaded code generated

by the compiler.

2 Enables a set of performance

tuning and heuristics in

compiler optimizations for

parallel code such as Windows

Threads, pthreads, and MPI

code, besides multithreaded

code generated by the

compiler.

#### **Default**

-opt-mem- For serial (non-parallel) compilation, a set of bandwidth0 performance tuning and heuristics in compiler or/Qopt-mem- optimizations is enabled that is optimal for serial bandwidth0 code.

-opt-membandwidth1 or/Qopt-membandwidth1 If you specify compiler option -parallel (Linux) or /Qparallel (Windows), -openmp (Linux) or /Qopenmp (Windows), or Cluster OpenMP option -cluster-openmp (Linux), a set of performance tuning and heuristics in compiler optimizations for multithreaded code generated by the compiler is enabled.

#### Description

This option enables performance tuning and heuristics that control memory bandwidth use among processors. It allows the compiler to be less aggressive with optimizations that might consume more bandwidth, so that the bandwidth can be well-shared among multiple processors for a parallel program. For values of n greater than 0, the option tells the compiler to enable a set of performance tuning and heuristics in compiler optimizations such as prefetching, privatization, aggressive code motion, and so forth, for reducing memory bandwidth pressure and balancing memory bandwidth traffic among threads. This option can improve performance for threaded or parallel applications on multiprocessors or multicore processors, especially when the applications are bounded by memory bandwidth.

#### **Alternate Options**

None

#### See Also

<u>parallel</u>, <u>Qparallel</u> compiler option openmp, <u>Qopenmp</u> compiler option

## opt-mod-versioning, Qopt-mod-versioning

Enables or disables versioning of modulo operations for certain types of operands.

#### **IDE Equivalent**

None

#### Architectures

IA-64 architecture

## **Syntax**

-no-opt-mod-versioning

Mac OS X: None

Windows: /Qopt-mod-versioning

/Qopt-mod-versioning-

## Arguments

None

#### Default

-no-opt-mod-versioning Versioning of modulo operations is

or/Qopt-mod-versioning- disabled.

#### Description

This option enables or disables versioning of modulo operations for certain types of operands. It is used for optimization tuning.

Versioning of modulo operations may improve performance for x mod y when modulus y is a power of 2.

#### **Alternate Options**

None

# opt-multi-version-aggressive, Qopt-multi-version-aggressive

Tells the compiler to use aggressive multi-versioning to check for pointer aliasing and scalar replacement.

#### **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64 architectures

## **Syntax**

Linux and Mac OS X: -opt-multi-version-aggressive

-no-opt-multi-version-aggressive

Windows: /Qopt-multi-version-aggressive

/Qopt-multi-version-aggressive-

#### **Arguments**

None

#### Default

-no-opt-multi-versionaggressive

or/Qopt-multi-version-

aggressive-

The compiler uses default heuristics when checking for pointer aliasing

and scalar replacement.

#### Description

This option tells the compiler to use aggressive multi-versioning to check for pointer aliasing and scalar replacement. This option may improve performance.

## **Alternate Options**

## None

# opt-prefetch, Qopt-prefetch

Enables or disables prefetch insertion optimization.

## **IDE Equivalent**

Windows: **Optimization > Prefetch Insertion** 

Linux: None

Mac OS X: Optimization > Enable Prefetch Insertion

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -opt-prefetch[=n]

-no-opt-prefetch

Windows: /Qopt-prefetch[:n]

/Qopt-prefetch-

#### Arguments

*n* Is the level of detail in the report. Possible values

are:

O Disables software prefetching.

This is the same as specifying

-no-opt-prefetch ( $\operatorname{Linux}$ 

and Mac OS X) or /Qopt-

prefetch- (Windows).

1 to 4 Enables different levels of

software prefetching. If you do

not specify a value for n, the

default is 2 on IA-32 and Intel® 64 architecture; the default is 3 on IA-64 architecture. Use lower values to reduce the amount of prefetching.

#### Default

IA-64 architecture: -opt-

prefetch

or/Qopt-prefetch

On IA-64 architecture, prefetch insertion optimization is enabled.

IA-32 architecture and Intel® 64

architecture:

-no-opt-prefetch

or/Qopt-prefetch-

On IA-32 architecture and Intel® 64 architecture, prefetch insertion optimization is disabled.

#### Description

This option enables or disables prefetch insertion optimization. The goal of prefetching is to reduce cache misses by providing hints to the processor about when data should be loaded into the cache.

On IA-64 architecture, this option is enabled by default if you specify option O1 or higher. To disable prefetching at these optimization levels, specify -no-opt-prefetch (Linux and Mac OS X) or /Qopt-prefetch- (Windows).

On IA-32 architecture and Intel® 64 architecture, this option enables prefetching when higher optimization levels are specified.

#### **Alternate Options**

Linux and Mac OS X: -prefetch (this is a deprecated option)

Windows: /Qprefetch (this is a deprecated option)

# opt-prefetch-initial-values, Qopt-prefetch-initial-values

Enables or disables prefetches that are issued before a loop is entered.

## **IDE Equivalent**

None

#### Architectures

IA-64 architecture

## **Syntax**

Linux: -opt-prefetch-initial-values

-no-opt-prefetch-initial-values

Mac OS X: None

Windows: /Qopt-prefetch-initial-values

/Qopt-prefetch-initial-values-

## **Arguments**

None

#### Default

-opt-prefetch-initial- Prefetches are issued before a loop

values is entered.

Or/Qopt-prefetch-initial-

values

#### Description

This option enables or disables prefetches that are issued before a loop is entered. These prefetches target the initial iterations of the loop.

When -O1 or higher is specified on IA-64 architecture, prefetches are issued before a loop is entered. To disable these prefetches, specify -no-opt-prefetch-initial-values (Linux) or /Qopt-prefetch-initial-values- (Windows).

<b>Alternate</b>	<b>Options</b>
------------------	----------------

None

# opt-prefetch-issue-excl-hint, Qopt-prefetch-issue-excl-hint

Determines whether the compiler issues prefetches for stores with exclusive hint.

## **IDE Equivalent**

None

#### Architectures

IA-64 architecture

## **Syntax**

Linux: -opt-prefetch-issue-excl-hint

-no-opt-prefetch-issue-excl-hint

Mac OS X: None

Windows: /Qopt-prefetch-issue-excl-hint

/Qopt-prefetch-issue-excl-hint-

## Arguments

None

#### Default

-no-opt-prefetch-issue- The compiler does not issue

excl-hint prefetches for stores with exclusive

or/Qopt-prefetch-issue- hint.

excl-hint-

## Description

This option determines whether the compiler issues prefetches for stores with exclusive hint. If option <code>-opt-prefetch-issue-excl-hint</code> (Linux) or

/Qopt-prefetch-issue-excl-hint (Windows) is specified, the prefetches will be issued if the compiler determines it is beneficial to do so.

When prefetches are issued for stores with exclusive-hint, the cache-line is in "exclusive-mode". This saves on cache-coherence traffic when other processors try to access the same cache-line. This feature can improve performance tuning.

## **Alternate Options**

None

## opt-prefetch-next-iteration, Qopt-prefetch-next-iteration

Enables or disables prefetches for a memory access in the next iteration of a loop.

## **IDE Equivalent**

None

#### **Architectures**

IA-64 architecture

#### **Syntax**

Linux: -opt-prefetch-next-iteration

-no-opt-prefetch-next-iteration

Mac OS X: None

Windows: /Qopt-prefetch-next-iteration

/Qopt-prefetch-next-iteration-

## **Arguments**

None

#### Default

-opt-prefetch-next- Prefetches are issued for a memory

iteration access in the next iteration of a

or/Qopt-prefetch-next- loop.

iteration

# Description

This option enables or disables prefetches for a memory access in the next iteration of a loop. It is typically used in a pointer-chasing loop.

When -O1 or higher is specified on IA-64 architecture, prefetches are issued for a memory access in the next iteration of a loop. To disable these prefetches, specify -no-opt-prefetch-next-iteration (Linux) or /Qopt-prefetch-next-iteration- (Windows).

#### **Alternate Options**

None

# opt-ra-region-strategy, Qopt-ra-region-strategy

Selects the method that the register allocator uses to partition each routine into regions.

# **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64 architectures

## Syntax

Linux and Mac OS X: -opt-ra-region-strategy[=keyword]

Windows: /Qopt-ra-region-strategy[:keyword]

## Arguments

keyword Is the method used for partitioning. Possible

## values are:

routine Creates a single region for

each routine.

block Partitions each routine into one

region per basic block.

trace Partitions each routine into one

region per trace.

region Partitions each routine into one

region per loop.

default The compiler determines which

method is used for partitioning.

#### Default

-opt-ra-region- The compiler determines which

strategy=default method is used for partitioning. This

or/Qopt-ra-region- is also the default if keyword is not

strategy:default specified.

## Description

This option selects the method that the register allocator uses to partition each routine into regions.

When setting default is in effect, the compiler attempts to optimize the tradeoff between compile-time performance and generated code performance.

This option is only relevant when optimizations are enabled (01 or higher).

## **Alternate Options**

None

#### See Also

# o compiler option

# opt-report, Qopt-report

Tells the compiler to generate an optimization report to stderr.

## **IDE Equivalent**

Windows: Diagnostics > Optimization Diagnostics Level

Linux: None

Mac OS X: None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -opt-report[n]

Windows: /Qopt-report[:n]

## Arguments

*n* Is the level of detail in the report. Possible values

are:

O Tells the compiler to generate

no optimization report.

1 Tells the compiler to generate

a report with the minimum level

of detail.

2 Tells the compiler to generate

a report with the medium level

of detail.

3 Tells the compiler to generate

a report with the maximum

## level of detail.

#### Default

-opt- If you do not specify n, the compiler generates a report

report 2 or with medium detail. If you do not specify the option on the

/Qopt- command line, the compiler does not generate an

report:2 optimization report.

## Description

This option tells the compiler to generate an optimization report to stderr.

## **Alternate Options**

None

## See Also

<u>opt-report-file</u>, <u>Qopt-report-file</u> compiler option Optimizing Applications: Optimizer Report Generation

## opt-report-file, Qopt-report-file

Specifies the name for an optimization report.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -opt-report-file=file

Windows: /Qopt-report-file:file

## Arguments

file Is the name for the optimization report.

Default

OFF No optimization report is generated.

## Description

This option specifies the name for an optimization report. If you use this option, you do not have to specify -opt-report (Linux and Mac OS X) or /Qopt-report (Windows).

# **Alternate Options**

None

#### See Also

opt-report, Qopt-report compiler option

Optimizing Applications: Optimizer Report Generation

# opt-report-help, Qopt-report-help

Displays the optimizer phases available for report generation.

# **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -opt-report-help

Windows: /Qopt-report-help

## Arguments

None

376

## Default

OFF No optimization reports are generated.

## Description

This option displays the optimizer phases available for report generation using - opt-report-phase (Linux and Mac OS X) or /Qopt-report-phase (Windows). No compilation is performed.

## **Alternate Options**

None

#### See Also

```
opt-report, Qopt-report compiler option
opt-report-phase, Qopt-report-phase compiler option
```

## opt-report-phase, Qopt-report-phase

Specifies an optimizer phase to use when optimization reports are generated.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -opt-report-phase=phase

Windows: /Qopt-report-phase:phase

# **Arguments**

phase Is the phase to generate reports for. Some of the

possible values are:

ipo	The Interprocedural Optimizer phase
hlo	The High Level Optimizer phase
hpo	The High Performance Optimizer phase
ilo	The Intermediate Language Scalar Optimizer phase
ecg	The Code Generator phase (Windows and Linux systems using IA-64 architecture only)
ecg_swp	The software pipelining component of the Code Generator phase (Windows and Linux systems using IA-64 architecture only)
pgo	The Profile Guided Optimization phase
all	All optimizer phases

## Default

OFF No optimization reports are generated.

# Description

This option specifies an optimizer phase to use when optimization reports are generated. To use this option, you must also specify <code>-opt-report</code> (Linux and Mac OS X) or <code>/Qopt-report</code> (Windows).

This option can be used multiple times on the same command line to generate reports for multiple optimizer phases.

When one of the logical names for optimizer phases is specified for phase, all reports from that optimizer phase are generated.

To find all phase possibilities, use option <code>-opt-report-help</code> (Linux and Mac OS X) or <code>/Qopt-report-help</code> (Windows).

## **Alternate Options**

None

#### See Also

opt-report, Qopt-report compiler option

# opt-report-routine, Qopt-report-routine

Tells the compiler to generate reports on the routines containing specified text.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -opt-report-routine=string

Windows: /Qopt-report-routine:string

#### Arguments

string Is the text (string) to look for.

#### Default

OFF No optimization reports are generated.

## Description

This option tells the compiler to generate reports on the routines containing specified text as part of their name.

## **Alternate Options**

None

#### See Also

opt-report, Qopt-report compiler option

# opt-streaming-stores, Qopt-streaming-stores

Enables generation of streaming stores for optimization.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64 architectures

# **Syntax**

Linux and Mac OS X: -opt-streaming-stores keyword

Windows: /Qopt-streaming-stores:keyword

## Arguments

*keyword* Specifies whether streaming stores are generated.

Possible values are:

always Enables generation of

streaming stores for

optimization. The compiler

optimizes under the

assumption that the application

is memory bound.

never Disables generation of

streaming stores for

optimization. Normal stores are

performed.

auto Lets the compiler decide which

instructions to use.

#### Default

-opt- The compiler decides whether to use streaming stores or streaming- normal stores.

stores auto

or/Qopt-

streaming-

stores:auto

#### Description

This option enables generation of streaming stores for optimization. This method stores data with instructions that use a non-temporal buffer, which minimizes memory hierarchy pollution.

For this option to be effective, the compiler must be able to generate SSE2 (or higher) instructions. For more information, see compiler option x or ax.

This option may be useful for applications that can benefit from streaming stores.

# Alternate Options

None

## See Also

ax, Qax compiler optionx, Qx compiler option

opt-mem-bandwidth, Qopt-mem-bandwidth, Qx compiler option

Optimizing Applications: Vectorization Support

## opt-subscript-in-range, Qopt-subscript-in-range

Determines whether the compiler assumes no overflows in the intermediate computation of subscript expressions in loops.

## IDE Equivalent

None

#### Architectures

IA-32, Intel® 64 architectures

## Syntax

Linux and Mac OS X: -opt-subscript-in-range

-no-opt-subscript-in-range

Windows: /Qopt-subscript-in-range

/Qopt-subscript-in-range-

## Arguments

None

#### Default

-no-opt- The compiler assumes overflows in the

subscript-in- intermediate computation of subscript expressions

range in loops.

or/Qopt-

subscript-in-

range-

#### Description

This option determines whether the compiler assumes no overflows in the

intermediate computation of subscript expressions in loops.

If you specify -opt-subscript-in-range (Linux and Mac OS X) or /Qopt-

subscript-in-range (Windows), the compiler ignores any data type

conversions used and it assumes no overflows in the intermediate computation

of subscript expressions. This feature can enable more loop transformations.

**Alternate Options** 

None

Example

The following shows an example where these options can be useful. m is declared as type integer(kind=8) (64-bits) and all other variables inside the subscript are declared as type integer(kind=4) (32-bits):

A[i + j + (n + k) \* m]

optimize

See O.

Os

Enables optimizations that do not increase code size and produces smaller code

size than O2.

**IDE Equivalent** 

Windows: Optimization > Favor Size or Speed

Linux: None

Mac OS X: None

Architectures

IA-32, Intel® 64, IA-64 architectures

**Syntax** 

383

Linux and Mac OS X: -Os

Windows: /Os

Arguments

None

Default

OFF Optimizations are made for code speed. However, if O1 is

specified, Os is the default.

Description

This option enables optimizations that do not increase code size and produces smaller code size than o2. It disables some optimizations that increase code size for a small speed benefit.

This option tells the compiler to favor transformations that reduce code size over transformations that produce maximum performance.

**Alternate Options** 

None

See Also

o compiler option

Ot compiler option

Ot

Enables all speed optimizations.

**IDE Equivalent** 

Windows: Optimization > Favor Size or Speed (/Ot, /Os)

Linux: None

Mac OS X: None

# Architectures IA-32, Intel® 64, IA-64 architectures Syntax

Linux and Mac OS X: None

Windows: /Ot

# Arguments

None

#### Default

/Ot Optimizations are made for code speed.

If od is specified, all optimizations are disabled. If old is specified, old is is the default.

## Description

This option enables all speed optimizations.

## **Alternate Options**

None

# See Also

o compiler option

Os compiler option

#### Ox

See O.

# fomit-frame-pointer, Oy

Determines whether EBP is used as a general-purpose register in optimizations.

## **IDE Equivalent**

Windows: **Optimization > Omit Frame Pointers** 

Linux: None

Mac OS X: Optimization > Provide Frame Pointer

#### Architectures

-f[no-]omit-frame-pointer: IA-32 architecture, Intel® 64 architecture /Oy[-]: IA-32 architecture

## **Syntax**

Linux and Mac OS X: -fomit-frame-pointer

-fno-omit-frame-pointer

Windows: /Oy

/Oy-

#### Arguments

None

#### Default

-fomit-frame- EBP is used as a general-purpose register in

pointer optimizations. However, on Linux\* and Mac OS X

or /Oy systems, the default is -fno-omit-frame-

pointer if option -00 or -g is specified. On

Windows\* systems, the default is /Oy- if option

/od is specified.

#### Description

These options determine whether EBP is used as a general-purpose register in optimizations. Options -fomit-frame-pointer and /Oy allow this use.

Options - fno-omit-frame-pointer and /Oy- disallow it.

Some debuggers expect EBP to be used as a stack frame pointer, and cannot produce a stack backtrace unless this is so. The -fno-omit-frame-pointer

and /Oy- options direct the compiler to generate code that maintains and uses

EBP as a stack frame pointer for all functions so that a debugger can still

produce a stack backtrace without doing the following:

• For -fno-omit-frame-pointer: turning off optimizations with -00

For /Oy-: turning off /O1, /O2, or /O3 optimizations

The -fno-omit-frame-pointer option is set when you specify option -00 or

the -g option. The -fomit-frame-pointer option is set when you specify

option -01, -02, or -03.

The /Oy option is set when you specify the /O1, /O2, or /O3 option. Option

/Oy- is set when you specify the /Od option.

Using the -fno-omit-frame-pointer or /Oy- option reduces the number of

available general-purpose registers by 1, and can result in slightly less efficient

code.

**Alternate Options** 

Linux and Mac OS X: -fp (this is a deprecated option)

Windows: None

p

Compiles and links for function profiling with gprof(1).

**IDE Equivalent** 

None

Architectures

IA-32, Intel® 64, IA-64 architectures

**Syntax** 

Linux and Mac OS X: -p

Windows:

None

Arguments

387

None

Default

OFF Files are compiled and linked without profiling.

# Description

This option compiles and links for function profiling with gprof(1).

## Alternate Options

Linux and Mac OS X: -pg (only available on systems using IA-32 architecture or Intel® 64 architecture), -qp (this is a <u>deprecated</u> option)

Windows: None

Ρ

See preprocess-only.

## pad, Qpad

Enables the changing of the variable and array memory layout.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -pad

-nopad

Windows: /Qpad

/Qpad-

## Arguments

None

#### Default

-nopad Variable and array memory layout is performed by default methods. or

/Qpad-

#### Description

This option enables the changing of the variable and array memory layout. This option is effectively not different from the align option when applied to structures and derived types. However, the scope of pad is greater because it applies also to common blocks, derived types, sequence types, and structures.

# **Alternate Options**

None

#### See Also

align compiler option

## pad-source, Qpad-source

Specifies padding for fixed-form source records.

## IDE Equivalent

Windows: Language > Pad Fixed Form Source Lines

Linux: None

Mac OS X: Language > Pad Fixed Form Source Lines

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

```
Linux and Mac OS X: -pad-source
```

-nopad-source

Windows: /pad-source

/nopad-source
/Qpad-source
/Qpad-source-

#### **Arguments**

None

#### Default

-nopad- Fixed-form source records are not padded.

source

or

/Qpad-

source-

#### Description

This option specifies padding for fixed-form source records. It tells the compiler that fixed-form source lines shorter than the statement field width are to be padded with spaces to the end of the statement field. This affects the interpretation of character and Hollerith literals that are continued across source records.

The default value setting causes a warning message to be displayed if a character or Hollerith literal that ends before the statement field ends is continued onto the next source record. To suppress this warning message, specify option – warn nousage (Linux and Mac OS X) or /warn:nousage (Windows).

Specifying pad-source or /Qpad-source can prevent warning messages associated with option -warn usage (Linux and Mac OS X) or /warn:usage (Windows).

# **Alternate Options**

None

## See Also

warn compiler option

# par-report, Qpar-report

Controls the diagnostic information reported by the auto-parallelizer.

## **IDE Equivalent**

Windows: Compilation Diagnostics > Auto-Parallelizer Diagnostic Level

Linux: None

Mac OS X: Diagnostics > Auto-Parallelizer Report

## Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -par-report[n]

Windows: /Qpar-report[n]

## Arguments

*n* Is a value denoting which diagnostic messages to

report. Possible values are:

Tells the auto-parallelizer to

report no diagnostic

information.

1 Tells the auto-parallelizer to

report diagnostic messages for

loops successfully auto-

parallelized. The compiler also issues a "LOOP AUTO-PARALLELIZED" message for parallel loops.

Tells the auto-parallelizer to report diagnostic messages for loops successfully and unsuccessfully auto-parallelized.

Tells the auto-parallelizer to report the same diagnostic messages specified by 2 plus additional information about any proven or assumed dependencies inhibiting autoparallelization (reasons for not parallelizing).

### Default

report1 If you do not specify n, the compiler displays diagnostic report1 messages for loops successfully auto-parallelized. If you or/Qpar- do not specify the option on the command line, the default report1 is to display no parallel disgnostic messages.

## Description

This option controls the diagnostic information reported by the auto-parallelizer (parallel optimizer). To use this option, you must also specify -parallel (Linux and Mac OS X) or /Qparallel (Windows).

If this option is specified on the command line, the report is sent to stdout.

On Windows systems, if this option is specified from within the IDE, the report is included in the build log if the Generate Build Logs option is selected.

# **Alternate Options**

None

# par-runtime-control, Qpar-runtime-control

Generates code to perform run-time checks for loops that have symbolic loop bounds.

## **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

```
Linux and Mac OS X: -par-runtime-control
```

-no-par-runtime-control

Windows: /Qpar-runtime-control

/Qpar-runtime-control-

## Arguments

None

### Default

```
-no-par- The compiler uses default heuristics when runtime-control checking loops.

or/Qpar-runtime-
control-
```

## Description

This option generates code to perform run-time checks for loops that have symbolic loop bounds.

If the granularity of a loop is greater than the parallelization threshold, the loop will be executed in parallel.

If you do not specify this option, the compiler may not parallelize loops with symbolic loop bounds if the compile-time granularity estimation of a loop can not ensure it is beneficial to parallelize the loop.

## **Alternate Options**

None

# par-schedule, Qpar-schedule

Lets you specify a scheduling algorithm or a tuning method for loop iterations.

## **IDE Equivalent**

None

### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -par-schedule-keyword[=n]

Windows: /Qpar-schedule-keyword[[:]n]

## Arguments

keyword Specifies the scheduling algorithm or tuning

method. Possible values are:

auto Lets the compiler or run-time

system determine the scheduling algorithm.

static Divides iterations into

contiguous pieces.

static- Divides iterations into even-

balanced sized chunks.

static- Divides iterations into even-

steal sized chunks, but allows

threads to steal parts of

chunks from neighboring

threads.

dynamic Gets a set of iterations

dynamically.

guided Specifies a minimum number

of iterations.

guided- Divides iterations by using

analytical exponential distribution or

dynamic distribution.

runtime Defers the scheduling decision

until run time.

Is the size of the chunk or the number of iterations

for each chunk. This setting can only be specified

for static, dynamic, and guided. For more

information, see the descriptions of each keyword

below.

### Default

n

static- Iterations are divided into even-sized chunks and the chunks balanced are assigned to the threads in the team in a round-robin fashion in the order of the thread number.

## Description

This option lets you specify a scheduling algorithm or a tuning method for loop iterations. It specifies how iterations are to be divided among the threads of the team.

This option affects performance tuning and can provide better performance during auto-parallelization.

Option	Description			
-par-schedule-auto <b>or</b> /Qpar-schedule-auto	Lets the compiler or run-time system determine the scheduling algorithm. Any possible mapping may occur for iterations to threads in the team.			
-par-schedule-static or /Qpar-schedule-static	Divides iterations into contiguous pieces (chunks) of size $n$ . The chunks are assigned to threads in the team in a round-robin fashion in the order of the thread number. Note that the last chunk to be assigned may have a smaller number of iterations.  If no $n$ is specified, the iteration space is divided into chunks that are approximately equal in size, and each thread is assigned at most one chunk.			
-par-schedule-static-balanced  Or /Qpar-schedule-static- balanced	Divides iterations into even-sized chunks. The chunks are assigned to the threads in the team in a round-robin fashion in the order of the thread number.			
-par-schedule-static-steal or /Qpar-schedule-static-steal	Divides iterations into even-sized chunks, but when a thread completes its chunk, it can steal parts of chunks			

Option	Description
	assigned to neighboring threads.  Each thread keeps track of L and U, which represent the lower and upper bounds of its chunks respectively.  Iterations are executed starting from the lower bound, and simultaneously, L is updated to represent the new lower bound.
-par-schedule-dynamic Or /Qpar-schedule-dynamic	Can be used to get a set of iterations dynamically. Assigns iterations to threads in chunks as the threads request them. The thread executes the chunk of iterations, then requests another chunk, until no chunks remain to be assigned.  As each thread finishes a piece of the iteration space, it dynamically gets the next set of iterations. Each chunk contains $n$ iterations, except for the last chunk to be assigned, which may have fewer iterations. If no $n$ is specified, the default is 1.
-par-schedule-guided or /Qpar-schedule-guided	Can be used to specify a minimum number of iterations. Assigns iterations to threads in chunks as the threads request them. The thread executes the chunk of iterations, then requests another chunk, until no chunks remain to be assigned.

Option	Description
	For a chunk of size 1, the size of each chunk is proportional to the number of unassigned iterations divided by the number of threads, decreasing to 1.  For an $n$ with value $k$ (greater than 1), the size of each chunk is determined in the same way with the restriction that the chunks do not contain fewer than $k$ iterations (except for the last chunk to be assigned, which may have fewer than $k$ iterations). If no $n$ is specified, the default is 1.
-par-schedule-guided- analytical or /Qpar-schedule- guided-analytical	Divides iterations by using exponential distribution or dynamic distribution. The method depends on run-time implementation. Loop bounds are calculated with faster synchronization and chunks are dynamically dispatched at run time by threads in the team.
-par-schedule-runtime or /Qpar-schedule-runtime	Defers the scheduling decision until run time. The scheduling algorithm and chunk size are then taken from the setting of environment variable OMP_SCHEDULE.

# **Alternate Options**

None

# par-threshold, Qpar-threshold

Sets a threshold for the auto-parallelization of loops.

## **IDE Equivalent**

Windows: **Optimization > Threshold For Auto-Parallelization** 

Linux: None

Mac OS X: Optimization > Threshold For Auto-Parallelization

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -par-threshold[n]

Windows: /Qpar-threshold[[:]n]

## Arguments

n

Is an integer whose value is the threshold for the auto-parallelization of loops. Possible values are 0 through 100.

If n is 0, loops get auto-parallelized always, regardless of computation work volume.

If n is 100, loops get auto-parallelized when performance gains are predicted based on the compiler analysis data. Loops get auto-parallelized only if profitable parallel execution is almost certain.

The intermediate 1 to 99 values represent the percentage probability for profitable speed-up. For example, n=50 directs the compiler to parallelize only if there is a 50% probability of the code speeding up if executed in parallel.

## Default

-par- Loops get auto-parallelized only if profitable

threshold100 parallel execution is almost certain. This is also the

or/Qpar- default if you do not specify n.

threshold100

# Description

This option sets a threshold for the auto-parallelization of loops based on the probability of profitable execution of the loop in parallel. To use this option, you must also specify <code>-parallel</code> (Linux and Mac OS X) or <code>/Qparallel</code> (Windows). This option is useful for loops whose computation work volume cannot be determined at compile-time. The threshold is usually relevant when the loop trip count is unknown at compile-time.

The compiler applies a heuristic that tries to balance the overhead of creating multiple threads versus the amount of work available to be shared amongst the threads.

## **Alternate Options**

None

# parallel, Qparallel

Tells the auto-parallelizer to generate multithreaded code for loops that can be safely executed in parallel.

## **IDE Equivalent**

Windows: **Optimization > Parallelization** 

Linux: None

Mac OS X: **Optimization > Parallelization** 

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -parallel

Windows: /Qparallel

## Arguments

None

## Default

OFF Multithreaded code is not generated for loops that can be safely executed in parallel.

## Description

This option tells the auto-parallelizer to generate multithreaded code for loops that can be safely executed in parallel.

To use this option, you must also specify option 02 or 03.



On Mac OS X systems, when you enable automatic parallelization, you must also set the <code>DYLD\_LIBRARY\_PATH</code> environment variable within Xcode or an error will be displayed.

# **Alternate Options**

None

## See Also

o compiler option

# pc, Qpc

Enables control of floating-point significand precision.

## **IDE Equivalent**

None

## **Architectures**

IA-32, Intel® 64 architectures

## **Syntax**

Linux and Mac OS X: -pcn

Windows: /Qpcn

## Arguments

*n* Is the floating-point significand precision. Possible

values are:

Rounds the significand to 24

bits (single precision).

Rounds the significand to 53

bits (double precision).

80 Rounds the significand to 64

bits (extended precision).

### Default

-pc80 On Linux\* and Mac OS\* X systems, the floating-point or/Qpc64 significand is rounded to 64 bits. On Windows\* systems, the floating-point significand is rounded to 53 bits.

## Description

This option enables control of floating-point significand precision.

Some floating-point algorithms are sensitive to the accuracy of the significand, or fractional part of the floating-point value. For example, iterative operations like division and finding the square root can run faster if you lower the precision with the this option.

Note that a change of the default precision control or rounding mode, for example, by using the -pc32 (Linux and Mac OS X) or /Qpc32 (Windows) option or by user intervention, may affect the results returned by some of the mathematical functions.

# **Alternate Options**

None

### See Also

Floating-point Operations: Floating-point Options Quick Reference

# pdbfile

Specifies that any debug information generated by the compiler should be saved to a program database file.

# IDE Equivalent

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: None

Windows: /pdbfile[:file]

/nopdbfile

# Arguments

*file* Is the name of the program database file.

## Default

/nopdbfile Debug information generated by the compiler is not saved

to a program database file.

## Description

This option specifies that any debug information generated by the compiler should be saved to a program database file. To use this option, you must also specify /debug:full (or the equivalent).

If *file* is not specified, the default file name used is the name of your file with an extension of .pdb.

The compiler places debug information in the object file if you specify /nopdbfile or omit both /pdbfile and /debug:full (or the equivalent).

## **Alternate Options**

None

## See Also

debug (Windows\*) compiler option

pg

See p.

## pie

Produces a position-independent executable on processors that support it.

# **IDE Equivalent**

None

### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux: -pie

Mac OS X: None

Windows: None

Arguments

None

Default

OFF The driver does not set up special run-time

libraries and the linker does not perform the

optimizations on executables.

# Description

This option produces a position-independent executable on processors that support it. It is both a compiler option and a linker option. When used as a compiler option, this option ensures the linker sets up run-time libraries correctly. Normally the object linked has been compiled with option <code>-fpie</code>. When you specify <code>-pie</code>, it is recommended that you specify the same options that were used during compilation of the object.

## **Alternate Options**

None

See Also

fpie compiler option

prec-div, Qprec-div

Improves precision of floating-point divides.

**IDE Equivalent** 

None

**Architectures** 

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -prec-div

-no-prec-div

Windows: /Qprec-div

/Qprec-div-

## Arguments

None

## Default

-prec-div The compiler uses this method for floating-point divides.

or/Qprec-

div

## Description

This option improves precision of floating-point divides. It has a slight impact on speed.

With some optimizations, such as -xSSE2 (Linux) or /QxSSE2 (Windows), the compiler may change floating-point division computations into multiplication by the reciprocal of the denominator. For example, A/B is computed as A \* (1/B) to improve the speed of the computation.

However, sometimes the value produced by this transformation is not as accurate as full IEEE division. When it is important to have fully precise IEEE division, use this option to disable the floating-point division-to-multiplication optimization. The result is more accurate, with some loss of performance. If you specify -no-prec-div (Linux and Mac OS X) or /Qprec-div- (Windows), it enables optimizations that give slightly less precise results than full IEEE division.

A	lte	rn	ate	Or	oti	ons
---	-----	----	-----	----	-----	-----

None

## See Also

Floating-point Operations: Floating-point Options Quick Reference

# prec-sqrt, Qprec-sqrt

Improves precision of square root implementations.

# IDE Equivalent

None

## Architectures

IA-32, Intel® 64 architectures

## **Syntax**

Linux and Mac OS X: -prec-sqrt

-no-prec-sqrt

Windows: /Qprec-sqrt

/Qprec-sqrt-

## Arguments

None

### Default

-no-prec- The compiler uses a faster but less precise

sqrt implementation of square root.

or /Qprec- Note that the default is -prec-sqrt or /Qprec-sqrt if

sqrt- any of the following options are specified: /Od, /Op, or

/Qprec on Windows systems; -00, -mp (or -

fltconsistency), or -mp1 on Linux and Mac OS X

systems.

## Description

This option improves precision of square root implementations. It has a slight impact on speed.

This option inhibits any optimizations that can adversely affect the precision of a square root computation. The result is fully precise square root implementations, with some loss of performance.

## **Alternate Options**

None

## preprocess-only

Causes the Fortran preprocessor to send output to a file.

## **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -preprocess-only

Windows: /preprocess-only

## Arguments

None

### Default

OFF Preprocessed source files are output to the compiler.

# Description

This option causes the Fortran preprocessor to send output to a file.

The source file is preprocessed by the Fortran preprocessor, and the result for each source file is output to a corresponding .i or .i90 file.

Note that the source file is not compiled.

## **Alternate Options**

Linux and Mac OS X: -P

Windows: /P

## print-multi-lib

Prints information about where system libraries should be found.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -print-multi-lib

Windows: None

## Arguments

None

### Default

OFF No information is printed unless the option is specified.

## Description

This option prints information about where system libraries should be found, but no compilation occurs. It is provided for compatibility with gcc.

None

# prof-data-order, Qprof-data-order

Enables or disables data ordering if profiling information is enabled.

# IDE Equivalent

None

## **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux: -prof-data-order

-no-prof-data-order

Mac OS X: None

Windows: /Qprof-data-order

/Qprof-data-order-

# Arguments

None

## Default

```
-no-prof- Data ordering is disabled.
```

data-order

or/Qprof-

data-

order-

# Description

This option enables or disables data ordering if profiling information is enabled. It controls the use of profiling information to order static program data items.

For this option to be effective, you must do the following:

• For instrumentation compilation, you must specify -prof-gen=globdata (Linux) or /Qprof-gen:globdata (Windows).

• For feedback compilation, you must specify -prof-use (Linux) or /Qprof-use (Windows). You must not use multi-file optimization by specifying options such as option -ipo (Linux) or /Qipo (Windows), or option -ipo-c (Linux) or /Qipo-c (Windows).

## **Alternate Options**

None

### See Also

```
<u>prof-gen, Qprof-gen</u> compiler option<u>prof-use, Qprof-use</u> compiler option<u>prof-func-order, Qprof-func-order</u> compiler option
```

## prof-dir, Qprof-dir

Specifies a directory for profiling information output files.

## **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -prof-dir dir

Windows: /Oprof-dir dir

### **Arguments**

dir

Is the name of the directory.

### Default

OFF Profiling output files are placed in the directory where the program is compiled.

## Description

This option specifies a directory for profiling information output files (\*.dyn and \*.dpi). The specified directory must already exist.

You should specify this option using the same directory name for both instrumentation and feedback compilations. If you move the .dyn files, you need to specify the new path.

## **Alternate Options**

None

## See Also

Floating-point Operations:

Profile-guided Optimization (PGO) Quick Reference

Coding Guidelines for Intel(R) Architectures

# prof-file, Qprof-file

Specifies an alternate file name for the profiling summary files.

## **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -prof-file file

Windows: /Qprof-file file

### Arguments

file Is the name of the profiling summary file.

### Default

OFF The profiling summary files have the file name pgopti.\*

## Description

This option specifies an alternate file name for the profiling summary files. The *file* is used as the base name for files created by different profiling passes. If you add this option to profmerge, the .dpi file will be named *file*.dpi instead of pgopti.dpi.

If you specify <code>-prof-genx</code> (Linux and Mac OS X) or <code>/Qprof-genx</code> (Windows) with this option, the .spi and .spl files will be named <code>file</code>.spi and <code>file</code>.spl instead of pgopti.spi and pgopti.spl.

If you specify -prof-use (Linux and Mac OS X) or /Qprof-use (Windows) with this option, the .dpi file will be named *file*.dpi instead of pgopti.dpi.

## **Alternate Options**

None

## See Also

prof-gen, Qprof-gen compiler option
prof-use, Qprof-use compiler option

Optimizing Applications:

Profile-guided Optimizations Overview

Coding Guidelines for Intel(R) Architectures

Profile an Application

# prof-func-groups

Enables or disables function grouping if profiling information is enabled.

## **IDE Equivalent**

None

### **Architectures**

IA-32, Intel® 64 architectures

## **Syntax**

Linux: -prof-func-groups

-no-prof-func-groups

Mac OS X: None

Windows: None

## Arguments

None

### Default

```
-no-prof-func- Function grouping is disabled. groups
```

## Description

This option enables or disables function grouping if profiling information is enabled.

A "function grouping" is a profiling optimization in which entire routines are placed either in the cold code section or the hot code section.

If profiling information is enabled by option <code>-prof-use</code>, option <code>-prof-func-groups</code> is set and function grouping is enabled. However, if you explicitly enable <code>-prof-func-order</code> (Linux) or <code>/Qprof-func-order</code> (Windows), function ordering is performed instead of function grouping.

If you want to disable function grouping when profiling information is enabled, specify -no-prof-func-groups.

To set the hotness threshold for function grouping, use option -prof-hotness-threshold (Linux) or /Qprof-hotness-threshold (Windows).

## **Alternate Options**

-func-groups (this is a deprecated option)

### See Also

```
prof-use, Qprof-use compiler option
prof-func-order, Qprof-func-order compiler option
prof-hotness-threshold, Qprof-hotness-threshold compiler option
```

# prof-func-order, **Qprof-func-order**

Enables or disables function ordering if profiling information is enabled.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux: -prof-func-order

-no-prof-func-order

Mac OS X: None

Windows: /Qprof-func-order

/Qprof-func-order-

## Arguments

None

## Default

```
-no-prof- Function ordering is disabled.

func-order

or/Qprof-

func-

order-
```

## Description

This option enables or disables function ordering if profiling information is enabled.

For this option to be effective, you must do the following:

- For instrumentation compilation, you must specify -prof-gen=srcpos
   (Linux) or /Qprof-gen:srcpos (Windows).
- For feedback compilation, you must specify -prof-use (Linux) or /Qprof-use (Windows). You must not use multi-file optimization by specifying options such as option -ipo (Linux) or /Qipo (Windows), or option -ipo-c (Linux) or /Qipo-c (Windows).

If you enable profiling information by specifying option <code>-prof-use</code> (Linux) or <code>/Qprof-use</code> (Windows), <code>-prof-func-groups</code> (Linux) and <code>/Qprof-func-groups</code> (Windows) are set and function grouping is enabled. However, if you explicitly enable <code>-prof-func-order</code> (Linux) or <code>/Qprof-func-order</code> (Windows), function ordering is performed instead of function grouping. On Linux\* systems, this option is only available for Linux linker 2.15.94.0.1, or later.

To set the hotness threshold for function grouping and function ordering, use option -prof-hotness-threshold (Linux) or /Qprof-hotness-threshold (Windows).

## **Alternate Options**

### None

The following example shows how to use this option on a Windows system:

```
ifort /Qprof-gen:globdata file1.f90 file2.f90 /exe:instrumented.exe
```

```
./instrumented.exe
ifort /Qprof-use /Qprof-func-order file1.f90 file2.f90
/exe:feedback.exe
```

The following example shows how to use this option on a Linux system:

### See Also

```
prof-hotness-threshold, Qprof-hotness-threshold compiler option
prof-gen, Qprof-gen compiler option
prof-use, Qprof-use compiler option
prof-data-order, Qprof-data-order compiler option
prof-func-groups compiler option
```

# prof-gen, Qprof-gen

Produces an instrumented object file that can be used in profile-guided optimization.

## **IDE Equivalent**

None

### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -prof-gen[=keyword]

-no-prof-gen

Windows: /Qprof-gen[:keyword]

/Qprof-gen-

## **Arguments**

keyword Specifies details for the instrumented file. Possible

values are:

default Produces an instrumented

object file. This is the same as specifying -prof-gen (Linux\* and Mac OS\* X) or /Qprof-gen (Windows\*) with no keyword.

srcpos

Produces an instrumented object file that includes extra source position information. This option is the same as option -prof-genx (Linux\* and Mac OS\* X) or /Qprof-genx (Windows\*), which are deprecated.

globdata

Produces an instrumented object file that includes information for global data layout.

## Default

-no- Profile generation is disabled.

profgen or
/Qprofgen-

## Description

This option produces an instrumented object file that can be used in profileguided optimization. It gets the execution count of each basic block.

If you specify keyword srcpos or globdata, a static profile information file (.spi) is created. These settings may increase the time needed to do a parallel build using -prof-gen, because of contention writing the .spi file.

These options are used in phase 1 of the Profile Guided Optimizer (PGO) to instruct the compiler to produce instrumented code in your object files in preparation for instrumented execution.

## **Alternate Options**

None

#### See Also

Optimizing Applications:

**Basic PGO Options** 

Example of Profile-Guided Optimization

# prof-genx, Qprof-genx

This is a deprecated option. See prof-gen keyword srcpos.

# prof-hotness-threshold, Qprof-hotness-threshold

Lets you set the hotness threshold for function grouping and function ordering.

## **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

**Linux**: -prof-hotness-threshold=n

Mac OS X: None

Windows: /Qprof-hotness-threshold:n

## Arguments

n

Is the hotness threshold. *n* is a percentage having a value between 0 and 100 inclusive. If you specify 0, there will be no hotness threshold setting in effect for function grouping and function ordering.

#### Default

OFF

The compiler's default hotness threshold setting of 10 percent is in effect for function grouping and function ordering.

## Description

This option lets you set the hotness threshold for function grouping and function ordering.

The "hotness threshold" is the percentage of functions in the application that should be placed in the application's hot region. The hot region is the most frequently executed part of the application. By grouping these functions together into one hot region, they have a greater probability of remaining resident in the instruction cache. This can enhance the application's performance.

For this option to take effect, you must specify option -prof-use (Linux) or /Qprof-use (Windows) and one of the following:

- On Linux systems: -prof-func-groups or -prof-func-order
- On Windows systems: /Qprof-func-order

# **Alternate Options**

None

### See Also

<u>prof-use</u>, <u>Qprof-use</u> compiler option<u>prof-func-groups</u> compiler option<u>prof-func-order</u>, <u>Qprof-func-order</u> compiler option

# prof-src-dir, Qprof-src-dir

Determines whether directory information of the source file under compilation is considered when looking up profile data records.

## **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -prof-src-dir

-no-prof-src-dir

Windows: /Oprof-src-dir

/Qprof-src-dir-

## Arguments

None

## Default

```
-prof- Directory information is used when looking up profile data src-dir records in the .dpi file.
```

or/Qprof-

src-dir

# Description

This option determines whether directory information of the source file under compilation is considered when looking up profile data records in the .dpi file. To use this option, you must also specify option <code>-prof-use</code> (Linux and Mac OS X) or <code>/Qprof-use</code> (Windows).

If the option is enabled, directory information is considered when looking up the profile data records within the .dpi file. You can specify directory information by using one of the following options:

- Linux and Mac OS X: -prof-src-root or -prof-src-root-cwd
- Windows: /Qprof-src-root or /Qprof-src-root-cwd

If the option is disabled, directory information is ignored and only the name of the file is used to find the profile data record.

Note that options <code>-prof-src-dir</code> (Linux and Mac OS X) and <code>/Qprof-src-dir</code> (Windows) control how the names of the user's source files get represented within the .dyn or .dpi files. Options <code>-prof-dir</code> (Linux and Mac OS X) and <code>/Qprof-dir</code> (Windows) specify the location of the .dyn or the .dpi files.

## **Alternate Options**

None

### See Also

```
prof-use, Qprof-use compiler option
prof-src-root, Qprof-src-root compiler option
prof-src-root-cwd, Qprof-src-root-cwd compiler option
```

## prof-src-root, Qprof-src-root

Lets you use relative directory paths when looking up profile data and specifies a directory as the base.

### **IDE Equivalent**

None

## Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

```
Linux and Mac OS X: -prof-src-root=dir
```

Windows: /Qprof-src-root:dir

Arguments

*dir* Is the base for the relative paths.

Default

OFF The setting of relevant options determines the path used when looking up profile data records.

## Description

This option lets you use relative directory paths when looking up profile data in .dpi files. It lets you specify a directory as the base. The paths are relative to a base directory specified during the -prof-gen (Linux and Mac OS X) or /Oprof-gen (Windows) compilation phase.

This option is available during the following phases of compilation:

- Linux and Mac OS X: -prof-gen and -prof-use phases
- Windows: /Qprof-gen and /Qprof-use phases

When this option is specified during the <code>-prof-gen</code> or <code>/Qprof-gen</code> phase, it stores information into the .dyn or .dpi file. Then, when .dyn files are merged together or the .dpi file is loaded, only the directory information below the root directory is used for forming the lookup key.

When this option is specified during the <code>-prof-use</code> or <code>/Qprof-use</code> phase, it specifies a root directory that replaces the root directory specified at the <code>-prof-gen</code> or <code>/Qprof-gen</code> phase for forming the lookup keys.

To be effective, this option or option <code>-prof-src-root-cwd</code> (Linux and Mac OS X) or <code>/Qprof-src-root-cwd</code> (Windows) must be specified during the <code>-prof-gen</code> or <code>/Qprof-gen</code> phase. In addition, if one of these options is not specified, absolute paths are used in the .dpi file.

## **Alternate Options**

## None

Consider the initial -prof-gen compilation of the source file c:\user1\feature\_foo\myproject\common\glob.f90:

```
ifort -prof-gen -prof-src-root=c:\user1\feature_foo\myproject -c
common\glob.f90
```

For the -prof-use phase, the file glob.f90 could be moved into the directory c:\user2\feature\_bar\myproject\common\glob.f90 and profile information would be found from the .dpi when using the following:

```
ifort -prof-use -prof-src-root=c:\user2\feature_bar\myproject -c
common\glob.f90
```

If you do not use option <code>-prof-src-root</code> during the <code>-prof-gen</code> phase, by default, the <code>-prof-use</code> compilation can only find the profile data if the file is compiled in the <code>c:\user1\feature\_foo\my\_project\common</code> directory.

#### See Also

```
prof-gen, Qprof-gen compiler option
prof-use, Qprof-use compiler option
prof-src-dir, Qprof-src-dir compiler option
prof-src-root-cwd, Qprof-src-root-cwd compiler option
```

## prof-src-root-cwd, Qprof-src-root-cwd

Lets you use relative directory paths when looking up profile data and specifies the current working directory as the base.

### **IDE Equivalent**

None

### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## Syntax

```
Linux and Mac OS X: -prof-src-root-cwd
```

Windows: /Qprof-src-root-cwd

## **Arguments**

None

### Default

OFF The setting of relevant options determines the path used when looking up profile data records.

## Description

This option lets you use relative directory paths when looking up profile data in .dpi files. It specifies the current working directory as the base. To use this option, you must also specify option -prof-use (Linux and Mac OS) or /Qprof-use (Windows).

This option is available during the following phases of compilation:

- Linux and Mac OS X: -prof-gen and -prof-use phases
- Windows: /Qprof-gen and /Qprof-use phases

When this option is specified during the <code>-prof-gen</code> or <code>/Qprof-gen</code> phase, it stores information into the .dyn or .dpi file. Then, when .dyn files are merged together or the .dpi file is loaded, only the directory information below the root directory is used for forming the lookup key.

When this option is specified during the <code>-prof-use</code> or <code>/Qprof-use</code> phase, it specifies a root directory that replaces the root directory specified at the <code>-prof-gen</code> or <code>/Qprof-gen</code> phase for forming the lookup keys.

To be effective, this option or option <code>-prof-src-root</code> (Linux and Mac OS X) or <code>/Qprof-src-root</code> (Windows) must be specified during the <code>-prof-gen</code> or <code>/Qprof-gen</code> phase. In addition, if one of these options is not specified, absolute paths are used in the .dpi file.

## **Alternate Options**

None

### See Also

```
prof-gen, Qprof-gen compiler option
prof-use, Qprof-use compiler option
prof-src-dir, Qprof-src-dir compiler option
prof-src-root, Qprof-src-root compiler option
```

# prof-use, Qprof-use

Enables the use of profiling information during optimization.

## **IDE Equivalent**

None

## **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -prof-use[=arg]

-no-prof-use

Windows: /Qprof-use[:arg]

/Qprof-use-

## Arguments

arg Specifies additional instructions. Possible values

are:

weighted Tells the profmerge utility to

apply a weighting to the .dyn

file values when creating

the .dpi file to normalize the

data counts when the training

runs have differentexecution

durations. This argument only

has an effect when the

compiler invokes the profmerge

utility to create the .dpi file. This argument does not have an effect if the .dpi file was previously created without weighting.

[no]merge Enables or disables automatic invocation of the profmerge utility. The default is merge. Note that you cannot specify both weighted and nomerge. If you try to specify both values, a warning will be displayed and nomerge takes precedence.

default

Enables the use of profiling information during optimization. The profmerge utility is invoked by default. This value is the same as specifying -profuse (Linux and Mac OS X) or /Oprof-use (Windows) with no argument.

## Default

Profiling information is not used during optimization. -no-

prof-

use **or** 

/Qprof-

use-

## Description

This option enables the use of profiling information (including function splitting and function grouping) during optimization. It enables option -fnsplit (Linux) or /Qfnsplit (Windows).

This option instructs the compiler to produce a profile-optimized executable and it merges available profiling output files into a pgopti.dpi file.

Note that there is no way to turn off function grouping if you enable it using this option.

To set the hotness threshold for function grouping and function ordering, use option -prof-hotness-threshold (Linux) or /Qprof-hotness-threshold (Windows).

## **Alternate Options**

None

#### See Also

<u>prof-hotness-threshold</u>, <u>Qprof-hotness-threshold</u> compiler option Optimizing Applications:

Basic PGO Options

Example of Profile-Guided Optimization

## ansi-alias, Qansi-alias

Tells the compiler to assume that the program adheres to Fortran Standard type aliasability rules.

#### **IDE Equivalent**

None

### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -ansi-alias

-no-ansi-alias

Windows: Qansi-alias

Qansi-alias-

### Arguments

None

#### Default

-ansi- Programs adhere to Fortran Standard type aliasability rules.

alias

or

/Qansi-

alias

## Description

This option tells the compiler to assume that the program adheres to type aliasability rules defined in the Fortran Standard.

For example, an object of type real cannot be accessed as an integer. For information on the rules for data types and data type constants, see "Data Types, Constants, and Variables" in the Language Reference.

This option directs the compiler to assume the following:

- Arrays are not accessed out of arrays' bounds.
- Pointers are not cast to non-pointer types and vice-versa.
- References to objects of two different scalar types cannot alias. For example, an object of type integer cannot alias with an object of type real or an object of type real cannot alias with an object of type double precision.

If your program adheres to the Fortran Standard type aliasability rules, this option enables the compiler to optimize more aggressively. If it doesn't adhere to these rules, then you should disable the option with -no-ansi-alias (Linux and Mac

OS X) or /Qansi-alias- (Windows) so the compiler does not generate incorrect code.

# **Alternate Options**

None

# auto, Qauto

See automatic.

# auto-scalar, Qauto-scalar

Causes scalar variables of intrinsic types INTEGER, REAL, COMPLEX, and LOGICAL that do not have the SAVE attribute to be allocated to the run-time stack.

# **IDE Equivalent**

Windows: Data > Local Variable Storage (/Qsave, /Qauto, /Qauto\_scalar)

Linux: None

Mac OS X: None

# Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -auto-scalar

Windows: /Qauto-scalar

# Arguments

None

### Default

-auto- Scalar variables of intrinsic types INTEGER, REAL,

complex, and LOGICAL that do not have the SAVE attribute or are allocated to the run-time stack. Note that if option

/Qauto-recursive, -openmp (Linux and Mac OS X), or /Qopenmp

scalar (Windows) is specified, the default is automatic.

### Description

This option causes allocation of scalar variables of intrinsic types INTEGER, REAL, COMPLEX, and LOGICAL to the run-time stack. It is as if they were declared with the AUTOMATIC attribute.

It does not affect variables that have the SAVE attribute (which include initialized locals) or that appear in an EQUIVALENCE statement or in a common block. This option may provide a performance gain for your program, but if your program depends on variables having the same value as the last time the routine was invoked, your program may not function properly. Variables that need to retain their values across subroutine calls should appear in a SAVE statement. You cannot specify option save, auto, or automatic with this option.



On Windows NT\* systems, there is a performance penalty for addressing a stack frame that is too large. This penalty may be incurred with <code>/automatic</code>, <code>/auto</code>, or <code>/Qauto</code> because arrays are allocated on the stack along with scalars. However, with <code>/Qauto-scalar</code>, you would have to have more than 32K bytes of local scalar variables before you incurred the performance penalty. <code>/Qauto-scalar</code> enables the compiler to make better choices about which variables should be kept in registers during program execution.

### **Alternate Options**

None

#### See Also

auto compiler option
save compiler option

# autodouble, Qautodouble

See <u>real-size</u>.

# ax, Qax

Tells the compiler to generate multiple, processor-specific auto-dispatch code paths for Intel processors if there is a performance benefit.

### **IDE Equivalent**

Windows: Code Generation > Add Processor-Optimized Code Path

Optimization > Generate Alternate Code Paths

Linux: None

Mac OS X: Code Generation > Add Processor-Optimized Code Path

#### **Architectures**

IA-32, Intel® 64 architectures

## **Syntax**

Linux and Mac OS X: -axprocessor

Windows: /Qaxprocessor

### Arguments

processor Indicates the processor for which code is

generated. The following descriptions refer to Intel® Streaming SIMD Extensions (Intel® SSE) and Supplemental Streaming SIMD Extensions

(Intel® SSSE). Possible values are:

SSE4.2 Can generate Intel® SSE4

Efficient Accelerated String and

Text Processing instructions supported by Intel® Core™ i7 processors. Can generate Intel® SSE4 Vectorizing Compiler and Media Accelerator, Intel® SSSE3, SSE3, SSE2, and SSE instructions and it can optimize for the Intel® Core™ processor family.

Vectorizing Compiler and
Media Accelerator instructions
for Intel processors. Can
generate Intel® SSSE3, SSE3,
SSE2, and SSE instructions
and it can optimize for Intel®
45nm Hi-k next generation
Intel® Core™
microarchitecture. This
replaces value S, which is
deprecated.

SSSE3 Can generate Intel® SSSE3,
SSE3, SSE2, and SSE
instructions for Intel processors
and it can optimize for the
Intel® Core™2 Duo processor
family. This replaces value T,
which is deprecated.

SSE3 Can generate Intel® SSE3,

SSE2, and SSE instructions for

Intel processors and it can

optimize for processors based

on Intel® Core™

microarchitecture and Intel

NetBurst® microarchitecture.

This replaces value P, which is

deprecated.

SSE2 Can generate Intel® SSE2 and

SSE instructions for Intel

processors, and it can optimize

for Intel® Pentium® 4

processors, Intel® Pentium® M

processors, and Intel® Xeon®

processors with Intel® SSE2.

This value is not available on

Mac OS X systems. This

replaces value N, which is

deprecated.

#### Default

OFF No auto-dispatch code is generated. Processor-specific code is generated and is controlled by the setting of compiler option -m (Linux), compiler option /arch (Windows), or compiler option -x (Mac OS\* X).

## Description

This option tells the compiler to generate multiple, processor-specific autodispatch code paths for Intel processors if there is a performance benefit. It also generates a baseline code path. The baseline code is usually slower than the specialized code.

The baseline code path is determined by the architecture specified by the -x (Linux and Mac OS X) or /Qx (Windows) option. While there are defaults for the -x or /Qx option that depend on the operating system being used, you can specify an architecture for the baseline code that is higher or lower than the default. The specified architecture becomes the effective minimum architecture for the baseline code path.

If you specify both the -ax and -x options (Linux and Mac OS X) or the /Qax and /Qx options (Windows), the baseline code will only execute on processors compatible with the processor type specified by the -x or /Qx option.

This option tells the compiler to find opportunities to generate separate versions of functions that take advantage of features of the specified Intel® processor. If the compiler finds such an opportunity, it first checks whether generating a processor-specific version of a function is likely to result in a performance gain. If this is the case, the compiler generates both a processor-specific version of a function and a baseline version of the function. At run time, one of the versions is chosen to execute, depending on the Intel processor in use. In this way, the program can benefit from performance gains on more advanced Intel processors, while still working properly on older processors.

You can use more than one of the processor values by combining them. For example, you can specify -axSSE4.1, SSSE3 (Linux and Mac OS X) or /QaxSSE4.1, SSSE3 (Windows). You cannot combine the old style, deprecated options and the new options. For example, you cannot specify -axSSE4.1, T (Linux and Mac OS X) or /QaxSSE4.1, T (Windows).

Previous values W and K are deprecated. The details on replacements are as follows:

Mac OS X systems: On these systems, there is no exact replacement for W or K. You can upgrade to the default option -msse3 (IA-32 architecture) or option -msse3 (Intel® 64 architecture).

Windows and Linux systems: The replacement for W is -msse2 (Linux) or

/arch:SSE2 (Windows). There is no exact replacement for K. However, on

Windows systems, /Qaxk is interpreted as /arch: IA32; on Linux systems,

-axK is interpreted as -mia32. You can also do one of the following:

• Upgrade to option -msse2 (Linux) or option /arch: SSE2 (Windows). This

will produce one code path that is specialized for Intel® SSE2. It will not run

on earlier processors

Specify the two option combination -mia32 -axSSE2 (Linux) or

/arch:IA32 /QaxSSE2 (Windows). This combination will produce an

executable that runs on any processor with IA-32 architecture but with an

additional specialized Intel® SSE2 code path.

The -ax and /Qax options enable additional optimizations not enabled with

option -m or option /arch.

# **Alternate Options**

None

#### See Also

x, Qx compiler option

m compiler option

arch compiler option

### Qchkstk

Enables stack probing when the stack is dynamically expanded at run-time.

## **IDE Equivalent**

Windows: Run-time > Enable Stack Check Upon Expansion

Linux: None

Mac OS X: None

#### **Architectures**

IA-64 architecture

436

## Syntax

Linux and Mac OS X: None

Windows: /Qchkstk

/Qchkstk-

## Arguments

None

### Default

/Qchkstk Stack probing is enabled when the stack is dynamically expanded at run-time.

## Description

This option enables stack probing when the stack is dynamically expanded at run-time.

It instructs the compiler to generate a call to \_chkstk. The call will probe the requested memory and detect possible stack overflow.

To cancel the call to chkstk, specify /Ochkstk-.

## **Alternate Options**

None

### common-args, Qcommon-args

See assume.

# complex-limited-range, Qcomplex-limited-range

Determines whether the use of basic algebraic expansions of some arithmetic operations involving data of type COMPLEX is enabled.

## **IDE Equivalent**

Windows: Floating point > Limit COMPLEX Range

Linux: None

Mac OS X: Floating point > Limit COMPLEX Range

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: -complex-limited-range

-no-complex-limited-range

Windows: /Qcomplex-limited-range

/Qcomplex-limited-range-

### Arguments

None

#### Default

-no-complex-limited- Basic algebraic expansions of some

range arithmetic operations involving data of type

or/Qcomplex-limited- COMPLEX are disabled.

range-

# Description

This option determines whether the use of basic algebraic expansions of some arithmetic operations involving data of type COMPLEX is enabled.

When the option is enabled, this can cause performance improvements in programs that use a lot of COMPLEX arithmetic. However, values at the extremes of the exponent range may not compute correctly.

# **Alternate Options**

None

## cpp, Qcpp

See fpp, Qfpp.

# d-lines, Qd-lines

Compiles debug statements.

## **IDE Equivalent**

Windows: Language > Compile Lines With D in Column 1

Linux: None

Mac OS X: Language > Compile Lines With D in Column 1

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -d-lines

-nod-lines

Windows: /d-lines

/nod-lines
/Qd-lines

## Arguments

None

#### Default

nod- Debug lines are treated as comment lines.

lines

## Description

This option compiles debug statements. It specifies that lines in fixed-format files that contain a D in column 1 (debug statements) should be treated as source code.

## **Alternate Options**

Linux and Mac OS X: -DD

Windows: None

# diag, Qdiag

Controls the display of diagnostic information.

### **IDE Equivalent**

Windows: **Diagnostics > Disable Specific Diagnostics** (/Qdiag-disable id)

Diagnostics > Level of Static Analysis (/Qdiag-enable[:sv1,sv2, sv3])

Linux: None

Mac OS X: Diagnostics > Disable Specific Diagnostics (-diag-disable id)

Diagnostics > Level of Static Analysis (-diag-enable [sv1,sv2, sv3])

## Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -diag-type diag-list

Windows: /Qdiag-type:diag-list

### Arguments

type Is an action to perform on diagnostics. Possible

values are:

enable Enables a diagnostic message

or a group of messages.

disable Disables a diagnostic message

or a group of messages.

error Tells the compiler to change

diagnostics to errors.

warning Tells the compiler to change

diagnostics to warnings.

remark Tells the compiler to change

diagnostics to remarks

(comments).

diag-list

Is a diagnostic group or ID value. Possible values

are:

driver Specifies diagnostic

messages issued by the

compiler driver.

vec Specifies diagnostic

messages issued by the

vectorizer.

par Specifies diagnostic

messages issued by the

auto-parallelizer (parallel

optimizer).

sv[n] Specifies diagnostic

messages issued by the

Static Verifier. *n* can be any of the following: 1, 2,

3. For more details on

these values, see below.

warn Specifies diagnostic

messages that have a

"warning" severity level.

error Specifies diagnostic

messages that have an

"error" severity level.

remark Specifies diagnostic

messages that are

remarks or comments.

cpu-dispatch Specifies the CPU

dispatch remarks for

diagnostic messages.

These remarks are

enabled by default. This

diagnostic group is only

available on IA-32

architecture and Intel® 64

architecture.

id[,id,...] Specifies the ID number of

one or more messages. If

you specify more than one

message number, they

must be separated by

commas. There can be no

intervening white space

between each id.

tag[,tag,...] Specifies the mnemonic

name of one or more

messages. If you specify

more than one mnemonic

name, they must be

separated by commas.

There can be no intervening white space between each tag.

#### Default

OFF

The compiler issues certain diagnostic messages by default.

### Description

This option controls the display of diagnostic information. Diagnostic messages are output to stderr unless compiler option <code>-diag-file</code> (Linux and Mac OS X) or <code>/Qdiag-file</code> (Windows) is specified.

When *diag-list* value "warn" is used with the Static Verifier (sv) diagnostics, the following behavior occurs:

- Option -diag-enable warn (Linux and Mac OS X) and /Qdiagenable:warn (Windows) enable all Static Verifier diagnostics except those that have an "error" severity level. They enable all Static Verifier warnings, cautions, and remarks.
- Option -diag-disable warn (Linux and Mac OS X) and /Qdiag-disable:warn (Windows) disable all Static Verifier diagnostics except those that have an "error" severity level. They suppress all Static Verifier warnings, cautions, and remarks.

The following table shows more information on values you can specify for diaglist item sv.

diag-list	Description
Item	

sv[n] The value of *n* for Static Verifier messages can be any of the following:

1 Produces the diagnostics with severity level set to all critical errors.

diag-list Item	Descrip	tion
	2	Produces the diagnostics with severity level set to all errors. This is the default if n is not specified.
	3	Produces the diagnostics with severity level set to all errors and warnings.

To control the diagnostic information reported by the vectorizer, use the -vec-report (Linux and Mac OS X) or /Qvec-report (Windows) option.

To control the diagnostic information reported by the auto-parallelizer, use the – par–report (Linux and Mac OS X) or /Qpar–report (Windows) option.

# **Alternate Options**

enable vec	Linux and Mac OS X: -vec-report Windows: /Qvec-report
disable vec	Linux and Mac OS X: -vec-report0 Windows: /Qvec-report0
enable par	Linux and Mac OS X: -par-report Windows: /Qpar-report
disable par	Linux and Mac OS X: -par-report0 Windows: /Qpar-report0

#### Example

The following example shows how to enable diagnostic IDs 117, 230 and 450:

```
-diag-enable 117,230,450 ! Linux and Mac OS X systems /Qdiag-enable:117,230,450 ! Windows systems
```

The following example shows how to change vectorizer diagnostic messages to warnings:

```
-diag-enable vec -diag-warning vec ! Linux and Mac OS X systems /Qdiag-enable:vec /Qdiag-warning:vec ! Windows systems
```

Note that you need to enable the vectorizer diagnostics before you can change them to warnings.

The following example shows how to disable all auto-parallelizer diagnostic messages:

```
-diag-disable par ! Linux and Mac OS X systems /Qdiag-disable:par ! Windows systems
```

The following example shows how to produce Static Verifier diagnostic messages for all critical errors:

```
-diag-enable svl ! Linux and Mac OS X systems /Qdiag-enable:svl ! Windows system
```

The following example shows how to cause Static Verifier diagnostics (and default diagnostics) to be sent to a file:

```
-diag-enable sv -diag-file=stat_ver_msg ! Linux and Mac OS X systems /Qdiag-enable:sv /Qdiag-file:stat_ver_msg ! Windows systems

Note that you need to enable the Static Verifier diagnostics before you can send them to a file. In this case, the diagnostics are sent to file stat_ver_msg.diag. If a file name is not specified, the diagnostics are sent to name-of-the-first-source-file.diag.
```

The following example shows how to change all diagnostic warnings and remarks to errors:

```
-diag-error warn, remark ! Linux and Mac OS X systems /Qdiag-error:warn, remark ! Windows systems
```

#### See Also

```
diag-dump, Qdiag-dump compiler option
diag-id-numbers, Qdiag-id-numbers compiler option
diag-file, Qdiag-file compiler option
par-report, Qpar-report compiler option
vec-report, Qvec-report compiler option
```

# diag-dump, Qdiag-dump

Tells the compiler to print all enabled diagnostic messages and stop compilation.

#### **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -diag-dump

Windows: /Qdiag-dump

### **Arguments**

None

#### Default

OFF The compiler issues certain diagnostic messages by default.

# Description

This option tells the compiler to print all enabled diagnostic messages and stop compilation. The diagnostic messages are output to stdout.

This option prints the enabled diagnostics from all possible diagnostics that the compiler can issue, including any default diagnostics.

If -diag-enable *diag-list* (Linux and Mac OS X) or /Qdiag-enable *diag-list* (Windows) is specified, the print out will include the *diag-list* diagnostics.

### **Alternate Options**

None

#### Example

The following example adds vectorizer diagnostic messages to the printout of default diagnostics:

```
-diag-enable vec -diag-dump ! Linux and Mac OS X systems /Qdiag-enable:vec /Qdiag-dump ! Windows systems
```

#### See Also

diag, Qdiag compiler option

# diag-enable sv-include, Qdiag-enable sv-include

Tells the Static Verifier to analyze include files and source files when issuing diagnostic messages.

### **IDE Equivalent**

Windows: **Diagnostics > Analyze Include Files** 

Linux: None

Mac OS X: Diagnostics > Analyze Include Files

### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -diag-enable sv-include

Windows: /Qdiag-enable sv-include

### Arguments

None

## Default

OFF The compiler issues certain diagnostic messages by default. If the Static Verifier is enabled, include files are not analyzed by default.

### Description

This option tells the Static Verifier to analyze include files and source files when issuing diagnostic messages. Normally, when Static Verifier diagnostics are enabled, only source files are analyzed.

To use this option, you must also specify <code>-diag-enable</code> sv (Linux and Mac OS X) or <code>/Qdiag-enable:sv</code> (Windows) to enable the Static Verifier diagnostics.

### **Alternate Options**

None

### Example

The following example shows how to cause include files to be analyzed as well as source files:

```
-diag-enable sv -diag-enable sv-include ! Linux and Mac OS systems /Qdiag-enable:sv /Qdiag-enable:sv-include ! Windows systems
```

In the above example, the first compiler option enables Static Verifier messages.

The second compiler option causes include files referred to by the source file to be analyzed also.

#### See Also

diag, Qdiag compiler option

# diag-error-limit, Qdiag-error-limit

Specifies the maximum number of errors allowed before compilation stops.

# IDE Equivalent

Windows: Compilation Diagnostics > Error Limit

Linux: None

Mac OS X: Compiler Diagnostics > Error Limit

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -diag-error-limitn

-no-diag-error-limit

Windows: /Qdiag-error-limit:n

/Odiag-error-limit-

#### Arguments

*n* Is the maximum number of error-level or fatal-level

compiler errors allowed.

#### Default

30

A maximum of 30 error-level and fatal-level messages are allowed.

### Description

This option specifies the maximum number of errors allowed before compilation stops. It indicates the maximum number of error-level or fatal-level compiler errors allowed for a file specified on the command line.

If you specify -no-diag-error-limit (Linux and Mac OS X) or /Qdiag-error-limit- (Windows) on the command line, there is no limit on the number of errors that are allowed.

If the maximum number of errors is reached, a warning message is issued and the next file (if any) on the command line is compiled.

## **Alternate Options**

Linux and Mac OS X: -error-limit and -noerror-limit

Windows: /error-limit and /noerror-limit

## diag-file, Qdiag-file

Causes the results of diagnostic analysis to be output to a file.

### **IDE Equivalent**

Windows: **Diagnostics > Diagnostics File** 

Linux: None

Mac OS X: Diagnostics > Diagnostics File

### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -diag-file[=file]

Windows: /Qdiag-file[:file]

## Arguments

file

Is the name of the file for output.

#### Default

OFF Diagnostic messages are output to stderr.

## Description

This option causes the results of diagnostic analysis to be output to a file. The file is placed in the current working directory.

If *file* is specified, the name of the file is *file.diag*. The file can include a file extension; for example, if *file.ext* is specified, the name of the file is *file.ext*. If *file* is not specified, the name of the file is *name-of-the-first-source-file.diag*. This is also the name of the file if the name specified for file conflicts with a source file name provided in the command line.



If you specify -diag-file (Linux and Mac OS X) or /Qdiag-file (Windows) and you also specify -diag-file-append (Linux and Mac OS X) or /Qdiag-file-append (Windows), the last option specified on the command line takes precedence.

### **Alternate Options**

None

### Example

The following example shows how to cause diagnostic analysis to be output to a file named my\_diagnostics.diag:

```
-diag-file=my_diagnostics ! Linux and Mac OS X systems /Qdiag-file:my_diagnostics ! Windows systems
```

#### See Also

diag, Qdiag compiler option

diag-file-append, Qdiag-file-append compiler option

# diag-file-append, Qdiag-file-append

Causes the results of diagnostic analysis to be appended to a file.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -diag-file-append[=file]

Windows: /Qdiag-file-append[:file]

### Arguments

file Is the name of the file to be appended to. It can include a path.

#### Default

OFF Diagnostic messages are output to stderr.

## Description

This option causes the results of diagnostic analysis to be appended to a file. If you do not specify a path, the driver will look for *file* in the current working directory.

If *file* is not found, then a new file with that name is created in the current working directory. If the name specified for file conflicts with a source file name provided in the command line. the name of the file is *name-of-the-first-source-file.diag*.



If you specify -diag-file-append (Linux and Mac OS X) or /Qdiag-file-append (Windows) and you also specify -diag-file (Linux and Mac OS X) or /Qdiag-file (Windows), the last option specified on the command line takes precedence.

# **Alternate Options**

None

# Example

The following example shows how to cause diagnostic analysis to be appended to a file named my\_diagnostics.txt:

```
-diag-file-append=my_diagnostics.txt ! Linux and Mac OS X systems /Qdiag-file-append:my_diagnostics.txt ! Windows systems
```

#### See Also

```
diag, Qdiag compiler option
diag-file, Qdiag-file compiler option
```

# diag-id-numbers, Qdiag-id-numbers

Determines whether the compiler displays diagnostic messages by using their ID number values.

#### **IDE** Equivalent

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

```
Linux and Mac OS X: -diag-id-numbers
```

-no-diag-id-numbers

Windows: /Odiag-id-numbers

/Qdiag-id-numbers-

## Arguments

None

## Default

-diag-id- The compiler displays diagnostic messages by

numbers using their ID number values.

or/Qdiag-id-

numbers

# Description

This option determines whether the compiler displays diagnostic messages by using their ID number values. If you specify -no-diag-id-numbers (Linux and Mac OS X) or /Qdiag-id-numbers- (Windows), mnemonic names are output for driver diagnostics only.

## **Alternate Options**

None

## See Also

diag, Qdiag compiler option

# diag-once, Qdiag-once

Tells the compiler to issue one or more diagnostic messages only once.

## **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -diag-onceid[,id,...]

Windows: /Qdiag-once:id[,id,...]

Arguments

id Is the ID number of the diagnostic message. If you specify more

than one message number, they must be separated by commas.

There can be no intervening white space between each *id*.

Default

OFF The compiler issues certain diagnostic messages by default.

Description

This option tells the compiler to issue one or more diagnostic messages only

once.

**Alternate Options** 

None

dps, Qdps

See altparam.

dyncom, Qdyncom

Enables dynamic allocation of common blocks at run time.

**IDE Equivalent** 

Windows: **Data > Dynamic Common Blocks** 

Linux: None

Mac OS X: Data > Dynamic Common Blocks

Architectures

IA-32, Intel® 64, IA-64 architectures

454

### **Syntax**

### **Arguments**

```
common1,common2,... Are the names of the common blocks to be dynamically allocated. The list of names must be within quotes.
```

#### Default

OFF Common blocks are not dynamically allocated at run time.

## Description

This option enables dynamic allocation of the specified common blocks at run time. For example, to enable dynamic allocation of common blocks a, b, and c at run time, use this syntax:

```
/Qdyncom "a,b,c" ! on Windows systems
-dyncom "a,b,c" ! on Linux and Mac OS X systems
```

The following are some limitations that you should be aware of when using this option:

- An entity in a dynamic common cannot be initialized in a DATA statement.
- Only named common blocks can be designated as dynamic COMMON.
- An entity in a dynamic common block must not be used in an EQUIVALENCE expression with an entity in a static common block or a DATA-initialized variable.

## **Alternate Options**

None

#### See Also

**Building Applications: Allocating Common Blocks** 

### **Qextend-source**

See <u>extend-source</u>.

## fast-transcendentals, Qfast-transcendentals

Enables the compiler to replace calls to transcendental functions with faster but less precise implementations.

### **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

### Syntax

Linux and Mac OS X: -fast-transcendentals

-no-fast-transcendentals

Windows: /Qfast-transcendentals

/Qfast-transcendentals-

#### Default

```
The default depends on the setting of -fp-model

transcendentals (Linux and Mac OS X) or /fp (Windows).

or /Qfast- The default is ON if default setting -fp-model

transcendentals fast or /fp:fast is in effect. However, if a value-
safe option such as -fp-model precise or

/fp:precise is specified, the default is OFF.
```

### Description

This option enables the compiler to replace calls to transcendental functions with implementations that may be faster but less precise.

It tells the compiler to perform certain optimizations on transcendental functions, such as replacing individual calls to sine in a loop with a single call to a less precise vectorized sine library routine.

This option has an effect only when specified with one of the following options:

- Windows\* OS: /fp:except or /fp:precise
- Linux\* OS and Mac OS\* X: -fp-model except or -fp-model precise
  You cannot use this option with option -fp-model strict (Linux and Mac OS
  X) or /fp:strict (Windows).

### **Alternate Options**

None

#### See Also

fp-model, fp compiler option

# fma, Qfma

Enables the combining of floating-point multiplies and add/subtract operations.

# IDE Equivalent

Windows: Floating Point > Contract Floating-Point Operations

Linux: None

Mac OS X: None

#### **Architectures**

IA-64 architecture

## Syntax

Linux: -fma

-no-fma

Mac OS X: None

Windows: /Qfma

/Qfma-

#### Arguments

None

#### Default

-fma Floating-point multiplies and add/subtract operations are

or/Qfma combined.

However, if you specify -mp (Linux), /Op (Windows),

/fp:strict (Windows), or -fp-model strict (Linux)

but do not explicitly specify -fma or /Qfma, the default is

-no-fma **or** /Qfma-.

# Description

This option enables the combining of floating-point multiplies and add/subtract operations.

It also enables the contraction of floating-point multiply and add/subtract operations into a single operation. The compiler contracts these operations whenever possible.

# **Alternate Options**

Linux: -IPF-fma (this is a deprecated option)

Windows: /QIPF-fma (this is a deprecated option)

#### See Also

fp-model, fp compiler option

Floating-point Operations: Floating-point Options Quick Reference

# falign-functions, Qfnalign

Tells the compiler to align functions on an optimal byte boundary.

## IDE Equivalent

None

#### Architectures

IA-32, Intel® 64 architectures

# Syntax

Linux and Mac OS X: -falign-functions[=n]

-fno-align-functions

Windows: /Qfnalign[:n]

/Qfnalign-

### Arguments

*n* Is the byte boundary for function alignment.

Possible values are 2 or 16.

#### Default

-fno- The compiler aligns functions on 2-byte boundaries. This

align- is the same as specifying -falign-functions=2

functions (Linux and Mac OS X) or /Qfnalign: 2 (Windows).

or

/Qfnalign-

### Description

This option tells the compiler to align functions on an optimal byte boundary. If you do not specify n, the compiler aligns the start of functions on 16-byte boundaries.

## **Alternate Options**

None

# fnsplit, Qfnsplit

Enables function splitting.

## **IDE Equivalent**

None

#### Architectures

/Qfnsplit[-]: IA-32 architecture, Intel® 64 architecture

-[no-]fnsplit: IA-64 architecture

## Syntax

Linux: -fnsplit

-no-fnsplit

Mac OS X: None

Windows: /Qfnsplit

/Qfnsplit-

### Arguments

None

### Default

-no-fnsplit Function splitting is not enabled unless -prof-

or/Qfnsplit- use (Linux) or /Qprof-use (Windows) is also

specified.

### Description

This option enables function splitting if <code>-prof-use</code> (Linux) or <code>/Qprof-use</code> (Windows) is also specified. Otherwise, this option has no effect. It is enabled automatically if you specify <code>-prof-use</code> or <code>/Qprof-use</code>. If you do not specify one of those options, the default is <code>-no-fnsplit</code> (Linux) or <code>/Qfnsplit-</code> (Windows), which disables function splitting but leaves function grouping enabled.

To disable function splitting when y	ou use -prof-use or /Qprof-use	, specify
-no-fnsplit <b>or</b> /Ofnsplit		

## **Alternate Options**

None

#### See Also

Optimizing Applications:

Basic PGO Options

Example of Profile-Guided Optimization

# fp-port, Qfp-port

Rounds floating-point results after floating-point operations.

## **IDE Equivalent**

Windows: Floating-Point > Round Floating-Point Results

Linux: None

Mac OS X: Floating-Point > Round Floating-Point Results

### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -fp-port

-no-fp-port

Windows: /Qfp-port

/Qfp-port-

# Arguments

None

### Default

-no-fp- The default rounding behavior depends on the compiler's

port code generation decisions and the precision parameters

or/Qfp- of the operating system.

port-

## Description

This option rounds floating-point results after floating-point operations. Rounding to user-specified precision occurs at assignments and type conversions. This has some impact on speed.

The default is to keep results of floating-point operations in higher precision. This provides better performance but less consistent floating-point results.

# **Alternate Options**

None

# fp-relaxed, Qfp-relaxed

Enables use of faster but slightly less accurate code sequences for math functions.

## **IDE Equivalent**

None

### Architectures

IA-64 architecture

## Syntax

Linux: -fp-relaxed

-no-fp-relaxed

Mac OS X: None

Windows: /Qfp-relaxed

/Qfp-relaxed-

#### Arguments

None

#### Default

```
-no-fp-relaxed Default code sequences are used for math or/Qfp-relaxed functions.
```

# Description

This option enables use of faster but slightly less accurate code sequences for math functions, such as divide and sqrt. When compared to strict IEEE\* precision, this option slightly reduces the accuracy of floating-point calculations performed by these functions, usually limited to the least significant digit.

This option also enables the performance of more aggressive floating-point transformations, which may affect accuracy.

# **Alternate Options**

Linux: -IPF-fp-relaxed (this is a <u>deprecated</u> option)

Windows: /QIPF-fp-relaxed (this is a deprecated option)

#### See Also

fp-model, fp compiler option

## fp-speculation, Qfp-speculation

Tells the compiler the mode in which to speculate on floating-point operations.

## **IDE Equivalent**

Windows: Floating Point > Floating-Point Speculation

Linux: None

Mac OS X: Floating Point > Floating-Point Speculation

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -fp-speculation=mode

Windows: /Qfp-speculation:mode

## Arguments

mode Is the mode for floating-point operations. Possible

values are:

fast Tells the compiler to speculate

on floating-point operations.

safe Tells the compiler to disable

speculation if there is a

possibility that the speculation

may cause a floating-point

exception.

strict Tells the compiler to disable

speculation on floating-point

operations.

off This is the same as specifying

strict.

## Default

-fp- The compiler speculates on floating-point speculation=fast operations. This is also the behavior when

or/Qfp- optimizations are enabled. However, if you specify

speculation: fast no optimizations (-00 on Linux; /Od on Windows),

the default is -fp-speculation=safe (Linux) or

/Qfp-speculation:safe (Windows).

## Description

This option tells the compiler the mode in which to speculate on floating-point operations.

## **Alternate Options**

None

#### See Also

Floating-point Operations: Floating-point Options Quick Reference

## fp-stack-check, Qfp-stack-check

Tells the compiler to generate extra code after every function call to ensure that the floating-point stack is in the expected state.

## **IDE Equivalent**

Windows: Floating-Point > Check Floating-point Stack

Linux: None

Mac OS X: Floating-Point > Check Floating-point Stack

## Architectures

IA-32, Intel® 64 architectures

## Syntax

Linux and Mac OS X: -fp-stack-check

Windows: /Qfp-stack-check

## Arguments

None

#### Default

OFF There is no checking to ensure that the floating-point (FP) stack

is in the expected state.

Description

This option tells the compiler to generate extra code after every function call to

ensure that the floating-point (FP) stack is in the expected state.

By default, there is no checking. So when the FP stack overflows, a NaN value is

put into FP calculations and the program's results differ. Unfortunately, the

overflow point can be far away from the point of the actual bug. This option

places code that causes an access violation exception immediately after an

incorrect call occurs, thus making it easier to locate these issues.

**Alternate Options** 

None

See Also

Floating-point Operations:

Checking the Floating-point Stack State

fpp, Qfpp

Runs the Fortran preprocessor on source files before compilation.

**IDE Equivalent** 

Windows: Preprocessor > Preprocess Source File

Linux: None

Mac OS X: Preprocessor > Preprocess Source File

Architectures

IA-32, Intel® 64, IA-64 architectures

**Syntax** 

Linux and Mac OS X: fpp[n]

fpp[="option"]

466

-nofpp Windows: /fpp[n]

/fpp[:"option"]

/nofpp /Qfpp[n]

/Qfpp[:"option"]

#### Arguments

*n* Deprecated. Tells the compiler whether to run the

preprocessor or not. Possible values are:

O Tells the compiler not to run

the preprocessor.

1, 2, or Tells the compiler to run the

3 preprocessor.

option Is a Fortran preprocessor (fpp) option; for

example, "-macro=no", which disables macro

expansion. The quotes are required. For a list of

fpp options, see Fortran Preprocessor Options.

#### Default

nofpp The Fortran preprocessor is not run on files before compilation.

#### Description

This option runs the Fortran preprocessor on source files before they are compiled.

If the option is specified with no argument, the compiler runs the preprocessor.

If 0 is specified for n, it is equivalent to nofpp. Note that argument n is deprecated.

We recommend you use option <code>Qoption,fpp,"option"</code> to pass fpp options to the Fortran preprocessor.

	Alternate Options		
	Linux and Mac OS X: -cpp		
Windows: /Qcpp			
	See Also		
	Fortran Preprocessor Options		
Qoption compiler option			
ftz, Qftz			
	Flushes denormal results to zero.		
	IDE Equivalent		
	Windows: (IA-32 and IA-64 architectures): Floating Point > Flush Denorm		
	Results to Zero		
	(Intel® 64 architecture): None		
	Linux: None		
	Mac OS X: Floating Point > Flush Denormal Results to Zero		
	Architectures		
	IA-32, Intel® 64, IA-64 architectures		
	Syntax		
Linux and Mac OS X: -ftz		-ftz	
		-no-ftz	
	Windows:	/Qftz	
		/Qftz-	
Arguments			
	None		

Default

Systems using IA-64 architecture: - On systems using IA-64 no-ftz or /Qftz-Systems using IA-32 architecture and Intel® 64 architecture: -ftz or systems using IA-32 architecture /Qftz

architecture, the compiler lets results gradually underflow. On and Intel® 64 architecture, denormal results are flushed to zero.

## Description

This option flushes denormal results to zero when the application is in the gradual underflow mode. It may improve performance if the denormal values are not critical to your application's behavior.

This option sets or resets the FTZ and the DAZ hardware flags. If FTZ is ON, denormal results from floating-point calculations will be set to the value zero. If FTZ is OFF, denormal results remain as is. If DAZ is ON, denormal values used as input to floating-point instructions will be treated as zero. If DAZ is OFF, denormal instruction inputs remain as is. Systems using IA-64 architecture have FTZ but not DAZ. Systems using Intel® 64 architecture have both FTZ and DAZ. FTZ and DAZ are not supported on all IA-32 architectures.

When -ftz (Linux and Mac OS X) or /Qftz (Windows) is used in combination with an SSE-enabling option on systems using IA-32 architecture (for example, xN or OxN), the compiler will insert code in the main routine to set FTZ and DAZ. When -ftz or /Oftz is used without such an option, the compiler will insert code to conditionally set FTZ/DAZ based on a run-time processor check. -noftz (Linux and Mac OS X) or /Oftz- (Windows) will prevent the compiler from inserting any code that might set FTZ or DAZ.

This option only has an effect when the main program is being compiled. It sets the FTZ/DAZ mode for the process. The initial thread and any threads subsequently created by that process will operate in FTZ/DAZ mode. Options -fpe0 and -fpe1 (Linux and Mac OS X) set -ftz. Options /fpe:0 and /fpe:1 (Windows) set /Oftz.

On systems using IA-64 architecture, optimization option O3 sets -ftz and /Qftz; optimization option O2 sets -no-ftz (Linux) and /Qftz- (Windows). On systems using IA-32 architecture and Intel® 64 architecture, every optimization option O level, except O0, sets -ftz and /Qftz. If this option produces undesirable results of the numerical behavior of your program, you can turn the FTZ/DAZ mode off by using -no-ftz or /Qftz- in the command line while still benefiting from the O3 optimizations.



Options -ftz and /Qftz are performance options. Setting these options does not guarantee that all denormals in a program are flushed to zero. They only cause denormals generated at run time to be flushed to zero.

#### **Alternate Options**

None

#### See Also

x, Qx compiler option

Floating-point Operations: Using the -fpe or /fpe Compiler Option Intrinsics Reference:

## global-hoist, Qglobal-hoist

Enables certain optimizations that can move memory loads to a point earlier in the program execution than where they appear in the source.

#### **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -global-hoist

-no-global-hoist

Windows: /Qglobal-hoist

/Qglobal-hoist-

## Arguments

None

#### Default

-global- Certain optimizations are enabled that can move memory

hoist loads.

or/Qglobal-

hoist

## Description

This option enables certain optimizations that can move memory loads to a point earlier in the program execution than where they appear in the source. In most cases, these optimizations are safe and can improve performance.

The -no-global-hoist (Linux and Mac OS X) or /Qglobal-hoist- (Windows) option is useful for some applications, such as those that use shared or dynamically mapped memory, which can fail if a load is moved too early in the execution stream (for example, before the memory is mapped).

#### **Alternate Options**

None

## QIA64-fr32

Disables use of high floating-point registers.

# **IDE Equivalent**

Windows: Floating Point > Disable Use of High Floating-Point Registers

# Intel® Fortran Compiler User and Reference Guides Linux: None Mac OS X: None Architectures IA-64 architecture **Syntax** Linux and Mac OS X: None Windows: /QIA64-fr32 Arguments None Default OFF Use of high floating-point registers is enabled. Description This option disables use of high floating-point registers. **Alternate Options** None **Qlfist** See rcd, Qrcd. inline-debug-info, Qinline-debug-info Produces enhanced source position information for inlined code. This is a

Produces enhanced source position information for inlined code. This is a deprecated option.

# **IDE Equivalent**

None

Architectures

IA-32, Intel® 64, IA-64 architectures

Syntax

Linux and Mac OS X: -inline-debug-info

/Qinline-debug-info Windows:

Arguments

None

Default

OFF No enhanced source position information is produced for inlined

code.

Description

This option produces enhanced source position information for inlined code. This

leads to greater accuracy when reporting the source location of any instruction. It

also provides enhanced debug information useful for function call traceback.

To use this option for debugging, you must also specify a debug enabling option,

such as -q (Linux) or /debug (Windows).

**Alternate Options** 

Linux and Mac OS X: -debug inline-debug-info

Windows: None

**Qinline-dllimport** 

Determines whether dllimport functions are inlined.

**IDE Equivalent** 

None

**Architectures** 

473

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: None

Windows: /Qinline-dllimport

/Qinline-dllimport-

## Arguments

None

## Default

```
/Qinline- The dllimport functions are inlined. dllimport
```

# Description

This option determines whether dllimport functions are inlined. To disable dllimport functions from being inlined, specify /Qinline-dllimport-.

# **Alternate Options**

None

# inline-factor, Qinline-factor

Specifies the percentage multiplier that should be applied to all inlining options that define upper limits.

# **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -inline-factor=n

-no-inline-factor

Windows: /Qinline-factor=n

/Qinline-factor-

## Arguments

*n* Is a positive integer specifying the percentage

value. The default value is 100 (a factor of 1).

#### Default

-no-inline- The compiler uses default heuristics for inline

factor routine expansion.

or/Qinline-

factor-

## Description

This option specifies the percentage multiplier that should be applied to all inlining options that define upper limits:

- -inline-max-size and /Qinline-max-size
- -inline-max-total-size and /Qinline-max-total-size
- -inline-max-per-routine and /Qinline-max-per-routine
- -inline-max-per-compile and /Oinline-max-per-compile

This option takes the default value for each of the above options and multiplies it by n divided by 100. For example, if 200 is specified, all inlining options that define upper limits are multiplied by a factor of 2. This option is useful if you do not want to individually increase each option limit.

If you specify -no-inline-factor (Linux and Mac OS X) or /Qinline-factor- (Windows), the following occurs:

 Every function is considered to be a small or medium function; there are no large functions.

- There is no limit to the size a routine may grow when inline expansion is performed.
- There is no limit to the number of times some routine may be inlined into a particular routine.
- There is no limit to the number of times inlining can be applied to a compilation unit.

To see compiler values for important inlining limits, specify compiler option – opt-report (Linux and Mac OS X) or /Qopt-report (Windows).



When you use this option to increase default limits, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

## **Alternate Options**

None

#### See Also

inline-max-size, Qinline-max-size compiler option
inline-max-total-size, Qinline-max-total-size compiler option
inline-max-per-routine, Qinline-max-per-routine compiler option
inline-max-per-compile, Qinline-max-per-compile compiler option
opt-report, Qopt-report compiler option

Optimizing Applications:

Developer Directed Inline Expansion of User Functions

Compiler Directed Inline Expansion of User Functions

## inline-forceinline, Qinline-forceinline

Specifies that an inline routine should be inlined whenever the compiler can do so.

# IDE Equivalent

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -inline-forceinline

Windows: /Qinline-forceinline

#### Default

OFF The compiler uses default heuristics for inline routine expansion.

## Description

This option specifies that a inline routine should be inlined whenever the compiler can do so. This causes the routines marked with an inline keyword or directive to be treated as if they were "forceinline".



Because C++ member functions whose definitions are included in the class declaration are considered inline functions by default, using this option will also make these member functions "forceinline" functions.

The "forceinline" condition can also be specified by using the directive cDEC\$ ATTRIBUTES FORCEINLINE.

To see compiler values for important inlining limits, specify compiler option – opt-report (Linux and Mac OS) or /Qopt-report (Windows).



When you use this option to change the meaning of inline to "forceinline", the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

#### **Alternate Options**

None

#### See Also

opt-report, Qopt-report compiler option

Optimizing Applications:

Developer Directed Inline Expansion of User Functions

Compiler Directed Inline Expansion of User Functions

## inline-max-per-compile, Qinline-max-per-compile

Specifies the maximum number of times inlining may be applied to an entire compilation unit.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -inline-max-per-compile=n

-no-inline-max-per-compile

Windows: /Qinline-max-per-compile=n

/Qinline-max-per-compile-

# Arguments

*n* Is a positive integer that specifies the number of

times inlining may be applied.

## Default

-no-inline-max-per- The compiler uses default heuristics for

compile inline routine expansion.

or/Qinline-max-percompile-

#### Description

This option the maximum number of times inlining may be applied to an entire compilation unit. It limits the number of times that inlining can be applied. For compilations using Interprocedural Optimizations (IPO), the entire compilation is a compilation unit. For other compilations, a compilation unit is a file.

If you specify -no-inline-max-per-compile (Linux and Mac OS X) or /Qinline-max-per-compile- (Windows), there is no limit to the number of times inlining may be applied to a compilation unit.

To see compiler values for important inlining limits, specify compiler option – opt-report (Linux and Mac OS X) or /Qopt-report (Windows).



When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

#### **Alternate Options**

None

### See Also

inline-factor, Qinline-factor compiler option
opt-report, Qopt-report compiler option

Optimizing Applications:

Developer Directed Inline Expansion of User Functions

Compiler Directed Inline Expansion of User Functions

## inline-max-per-routine, Qinline-max-per-routine

Specifies the maximum number of times the inliner may inline into a particular routine.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -inline-max-per-routine=n

-no-inline-max-per-routine

Windows: /Qinline-max-per-routine=n

/Qinline-max-per-routine-

## Arguments

*n* Is a positive integer that specifies the maximum

number of times the inliner may inline into a

particular routine.

#### Default

-no-inline-max-per- The compiler uses default heuristics for

routine inline routine expansion.

or/Qinline-max-per-

routine-

## Description

This option specifies the maximum number of times the inliner may inline into a particular routine. It limits the number of times that inlining can be applied to any routine.

If you specify -no-inline-max-per-routine (Linux and Mac OS X) or /Qinline-max-per-routine- (Windows), there is no limit to the number of times some routine may be inlined into a particular routine.

To see compiler values for important inlining limits, specify compiler option – opt-report (Linux and Mac OS X) or /Qopt-report (Windows).

To see compiler values for important inlining limits, specify compiler option – opt-report (Linux and Mac OS X) or /Qopt-report (Windows).



When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

#### **Alternate Options**

None

#### See Also

<u>inline-factor</u>, <u>Qinline-factor</u> compiler option <u>opt-report</u>, <u>Qopt-report</u> compiler option

Optimizing Applications:

Developer Directed Inline Expansion of User Functions

Compiler Directed Inline Expansion of User Functions

# inline-max-size, Qinline-max-size

Specifies the lower limit for the size of what the inliner considers to be a large routine.

#### **IDE Equivalent**

None

## Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -inline-max-size=n

-no-inline-max-size

Windows: /Qinline-max-size=n

/Qinline-max-size-

## Arguments

*n* Is a positive integer that specifies the minimum

size of what the inliner considers to be a large

routine.

#### Default

-no-inline-max-size The compiler uses default heuristics for or/Qinline-max-size inline routine expansion.

#### Description

This option specifies the lower limit for the size of what the inliner considers to be a large routine (a function or subroutine). The inliner classifies routines as small, medium, or large. This option specifies the boundary between what the inliner considers to be medium and large-size routines.

The inliner prefers to inline small routines. It has a preference against inlining large routines. So, any large routine is highly unlikely to be inlined.

If you specify -no-inline-max-size (Linux and Mac OS X) or /Qinline-max-size- (Windows), there are no large routines. Every routine is either a small or medium routine.

To see compiler values for important inlining limits, specify compiler option – opt-report (Linux and Mac OS X) or /Qopt-report (Windows).

To see compiler values for important inlining limits, specify compiler option – opt-report (Linux and Mac OS X) or /Qopt-report (Windows).



When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

## **Alternate Options**

None

#### See Also

```
<u>inline-min-size</u>, <u>Qinline-min-size</u> compiler option

<u>inline-factor</u>, <u>Qinline-factor</u> compiler option

<u>opt-report</u>, <u>Qopt-report</u> compiler option
```

Optimizing Applications:

Developer Directed Inline Expansion of User Functions

Compiler Directed Inline Expansion of User Functions

# inline-max-total-size, Qinline-max-total-size

Specifies how much larger a routine can normally grow when inline expansion is performed.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -inline-max-total-size=n

-no-inline-max-total-size

Windows: /Qinline-max-total-size=n

/Qinline-max-total-size-

#### **Arguments**

n

Is a positive integer that specifies the permitted increase in the routine's size when inline expansion is performed.

#### Default

```
-no-inline-max-total- The compiler uses default heuristics for size inline routine expansion.

or/Qinline-max-total-
size-
```

## Description

This option specifies how much larger a routine can normally grow when inline expansion is performed. It limits the potential size of the routine. For example, if 2000 is specified for n, the size of any routine will normally not increase by more than 2000.

If you specify -no-inline-max-total-size (Linux and Mac OS X) or /Qinline-max-total-size- (Windows), there is no limit to the size a routine may grow when inline expansion is performed.

To see compiler values for important inlining limits, specify compiler option – opt-report (Linux and Mac OS X) or /Qopt-report (Windows).

To see compiler values for important inlining limits, specify compiler option – opt-report (Linux and Mac OS X) or /Qopt-report (Windows).



When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

## **Alternate Options**

## None

#### See Also

<u>inline-factor</u>, <u>Qinline-factor</u> compiler option <u>opt-report</u>, <u>Qopt-report</u> compiler option

Optimizing Applications:

Developer Directed Inline Expansion of User Functions

Compiler Directed Inline Expansion of User Functions

## inline-min-size, Qinline-min-size

Specifies the upper limit for the size of what the inliner considers to be a small routine.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -inline-min-size=n

-no-inline-min-size

Windows: /Qinline-min-size=n

/Qinline-min-size-

## Arguments

*n* Is a positive integer that specifies the maximum

size of what the inliner considers to be a small

routine.

#### Default

```
-no-inline-min-size The compiler uses default heuristics for or/Qinline-min-size inline routine expansion.
```

### Description

This option specifies the upper limit for the size of what the inliner considers to be a small routine (a function or subroutine). The inliner classifies routines as small, medium, or large. This option specifies the boundary between what the inliner considers to be small and medium-size routines.

The inliner has a preference to inline small routines. So, when a routine is smaller than or equal to the specified size, it is very likely to be inlined.

If you specify -no-inline-min-size (Linux and Mac OS X) or /Qinline-min-size- (Windows), there is no limit to the size of small routines. Every routine is a small routine; there are no medium or large routines.

To see compiler values for important inlining limits, specify compiler option - opt-report (Linux and Mac OS X) or /Qopt-report (Windows).

To see compiler values for important inlining limits, specify compiler option - opt-report (Linux and Mac OS X) or /Qopt-report (Windows).



When you use this option to increase the default limit, the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

#### **Alternate Options**

None

#### See Also

<u>inline-min-size</u>, <u>Qinline-min-size</u> compiler option <u>opt-report</u>, <u>Qopt-report</u> compiler option Optimizing Applications: Developer Directed Inline Expansion of User Functions

486

Compiler Directed Inline Expansion of User Functions

#### Qinstall

Specifies the root directory where the compiler installation was performed.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -Qinstalldir

Windows: None

## Arguments

dir Is the root directory where the installation was

performed.

## Default

OFF The default root directory for compiler installation is searched for the compiler.

## Description

This option specifies the root directory where the compiler installation was performed. It is useful if you want to use a different compiler or if you did not use the ifortvars shell script to set your environment variables.

## **Alternate Options**

None

# minstruction, Qinstruction

Determines whether MOVBE instructions are generated for Intel processors.

# **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64 architectures

## **Syntax**

Linux and Mac OS X: -minstruction=[no]movbe

Windows: /Qinstruction:[no]movbe

#### Arguments

None

#### Default

-minstruction=movbe The compiler generates MOVBE or/Qinstruction:movbe instructions for Intel® Atom™ processors.

#### Description

This option determines whether MOVBE instructions are generated for Intel processors. To use this option, you must also specify -xSSE3\_ATOM (Linux and Mac OS X) or /QxSSE3\_ATOM (Windows).

If -minstruction=movbe or /Qinstruction:movbe is specified, the
following occurs:

- MOVBE instructions are generated that are specific to the Intel® Atom™
  processor.
- The options are ON by default when -xSSE3\_ATOM or /QxSSE3\_ATOM is specified.

 Generated executables can only be run on Intel® Atom™ processors or processors that support Intel® Streaming SIMD Extensions 3 (Intel® SSE3) and MOVBE.

If -minstruction=nomovbe or /Qinstruction:nomovbe is specified, the
following occurs:

- The compiler optimizes code for the Intel® Atom™ processor, but it does not generate MOVBE instructions.
- Generated executables can be run on non-Intel® Atom™ processors that support Intel® SSE3.

# **Alternate Options**

None

#### See Also

x, Qx compiler option

# finstrument-functions, Qinstrument-functions

Determines whether routine entry and exit points are instrumented.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

#### Syntax

Linux and Mac OS X: -finstrument-functions

-fno-instrument-functions

Windows: /Qinstrument-functions

/Qinstrument-functions-

## Arguments

#### None

#### Default

```
-fno- Routine entry and exit points are not instrumented.
instrument-
functions

Or/Qinstrument-
functions-
```

#### Description

This option determines whether routine entry and exit points are instrumented. It may increase execution time.

The following profiling functions are called with the address of the current routine and the address of where the routine was called (its "call site"):

- This function is called upon routine entry:
- On IA-32 architecture and Intel® 64 architecture:

```
void __cyg_profile_func_enter (void *this_fn,
void *call_site);
```

o On IA-64 architecture:

```
void __cyg_profile_func_enter (void **this_fn,
void *call_site);
```

- This function is called upon routine exit:
- On IA-32 architecture and Intel® 64 architecture:

```
void __cyg_profile_func_exit (void *this_fn,
void *call_site);
```

On IA-64 architecture:

```
void __cyg_profile_func_exit (void **this_fn,
void *call_site);
```

On IA-64 architecture, the additional de-reference of the function pointer argument is required to obtain the routine entry point contained in the first word of the routine descriptor for indirect routine calls. The descriptor is documented in the Intel® Itanium® Software Conventions and Runtime Architecture Guide, section 8.4.2. You can find this design guide at web site <a href="http://www.intel.com">http://www.intel.com</a>

These functions can be used to gather more information, such as profiling information or timing information. Note that it is the user's responsibility to provide these profiling functions.

If you specify -finstrument-functions (Linux and Mac OS X) or /Qinstrument-functions (Windows), routine inlining is disabled. If you specify -fno-instrument-functions or /Qinstrument-functions-, inlining is not disabled.

This option is provided for compatibility with gcc.

# **Alternate Options**

None

# ip, Qip

Determines whether additional interprocedural optimizations for single-file compilation are enabled.

# IDE Equivalent

Windows: **Optimization > Interprocedural Optimization** 

Linux: None

Mac OS X: None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

#### Syntax

Linux and Mac OS X: -ip

-no-ip

Windows: /Qip

/Qip-

#### **Arguments**

None

#### Default

OFF Some limited interprocedural optimizations occur, including inline function expansion for calls to functions defined within the current source file. These optimizations are a subset of full intra-file interprocedural optimizations. Note that this setting is not the same as -no-ip (Linux and Mac OS X) or /Qip- (Windows).

## Description

This option determines whether additional interprocedural optimizations for single-file compilation are enabled.

Options -ip (Linux and Mac OS X) and /Qip (Windows) enable additional interprocedural optimizations for single-file compilation.

Options -no-ip (Linux and Mac OS X) and /Qip- (Windows) may not disable inlining. To ensure that inlining of user-defined functions is disabled, specify -inline-level=0or-fno-inline (Linux and Mac OS X), or specify /Ob0 (Windows).

## **Alternate Options**

None

#### See Also

finline-functions compiler option

## ip-no-inlining, Qip-no-inlining

Disables full and partial inlining enabled by interprocedural optimization options.

# **IDE Equivalent**

None

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -ip-no-inlining

Windows: /Qip-no-inlining

## Arguments

None

#### Default

OFF Inlining enabled by interprocedural optimization options is performed.

## Description

This option disables full and partial inlining enabled by the following interprocedural optimization options:

- On Linux and Mac OS X systems: -ip or -ipo
- On Windows systems: /Qip, /Qipo, or /Ob2

It has no effect on other interprocedural optimizations.

On Windows systems, this option also has no effect on user-directed inlining specified by option /0b1.

## **Alternate Options**

None

# ip-no-pinlining, Qip-no-pinlining

Disables partial inlining enabled by interprocedural optimization options.

## **IDE Equivalent**

None

IA-32, Intel® 64 architectures

# Syntax

Linux and Mac OS X: -ip-no-pinlining

Windows: /Qip-no-pinlining

# Arguments

None

#### Default

OFF Inlining enabled by interprocedural optimization options is performed.

## Description

This option disables partial inlining enabled by the following interprocedural optimization options:

- On Linux and Mac OS X systems: -ip or -ipo
- On Windows systems: /Qip or /Qipo

It has no effect on other interprocedural optimizations.

#### **Alternate Options**

None

## IPF-fit-eval-method0, QIPF-fit-eval-method0

Tells the compiler to evaluate the expressions involving floating-point operands in the precision indicated by the variable types declared in the program. This is a deprecated option.

## **IDE Equivalent**

None

#### IA-64 architecture

## **Syntax**

Linux: -IPF-flt-eval-method0

Mac OS X: None

Windows: /QIPF-flt-eval-method0

## Arguments

None

#### Default

OFF Expressions involving floating-point operands are evaluated by default rules.

# Description

This option tells the compiler to evaluate the expressions involving floating-point operands in the precision indicated by the variable types declared in the program. By default, intermediate floating-point expressions are maintained in higher precision.

The recommended method to control the semantics of floating-point calculations is to use option -fp-model (Linux) or /fp (Windows).

Instead of using -IPF-flt-eval-method0 (Linux) or /QIPF-flt-evalmethod0 (Windows), you can use -fp-model source (Linux) or /fp:source
(Windows).

#### **Alternate Options**

None

### See Also

fp-model, fp compiler option

IPF-fltacc, QIPF-fltacc

Disables optimizations that affect floating-point accuracy. This is a deprecated option.

# IDE Equivalent

Windows: Floating Point > Floating-Point Accuracy

Linux: None

Mac OS X: None

#### Architectures

IA-64 architecture

## Syntax

Linux: -IPF-fltacc

-no-IPF-fltacc

Mac OS X: None

Windows: /QIPF-fltacc

/QIPF-fltacc-

## Arguments

None

#### Default

-no-IPF-fltacc Optimizations are enabled that affect floating-point

or/QIPF-fltacc- accuracy.

## Description

This option disables optimizations that affect floating-point accuracy.

If the default setting is used, the compiler may apply optimizations that reduce floating-point accuracy.

You can use this option to improve floating-point accuracy, but at the cost of disabling some optimizations.

The recommended method to control the semantics of floating-point calculations is to use option -fp-model (Linux) or /fp (Windows).

Instead of using -IPF-fltacc (Linux) or /QIPF-fltacc (Windows), you can
use -fp-model precise (Linux) or /fp:precise (Windows).

Instead of using -no-IPF-fltacc (Linux) or /QIPF-fltacc- (Windows), you
can use -fp-model fast (Linux) or /fp:fast (Windows).

## **Alternate Options**

None

#### See Also

fp-model, fp compiler option

IPF-fma, QIPF-fma

See fma, Qfma.

# IPF-fp-relaxed, QIPF-fp-relaxed

See fp-relaxed, Qfp-relaxed.

## ipo, Qipo

Enables interprocedural optimization between files.

## **IDE Equivalent**

Windows: **Optimization > Interprocedural Optimization** 

General > Whole Program Optimization

Linux: None

Mac OS X: None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -ipo[n]

Windows: /Qipo[n]

Arguments

*n* Is an optional integer that specifies the number of

object files the compiler should create. The integer

must be greater than or equal to 0.

**Default** 

OFF Multifile interprocedural optimization is not enabled.

Description

This option enables interprocedural optimization between files. This is also called multifile interprocedural optimization (multifile IPO) or Whole Program Optimization (WPO).

When you specify this option, the compiler performs inline function expansion for calls to functions defined in separate files.

You cannot specify the names for the files that are created.

If n is 0, the compiler decides whether to create one or more object files based on an estimate of the size of the application. It generates one object file for small applications, and two or more object files for large applications.

If n is greater than 0, the compiler generates n object files, unless n exceeds the number of source files (m), in which case the compiler generates only m object files.

If you do not specify n, the default is 0.

**Alternate Options** 

None

See Also

Optimizing Applications:

498

Interprocedural Optimization (IPO) Quick Reference Interprocedural Optimization (IPO) Overview Using IPO

# ipo-c, Qipo-c

Tells the compiler to optimize across multiple files and generate a single object file.

# **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: -ipo-c

Windows: /Qipo-c

### Arguments

None

#### Default

OFF The compiler does not generate a multifile object file.

### Description

This option tells the compiler to optimize across multiple files and generate a single object file (named ipo\_out.o on Linux and Mac OS X systems; ipo\_out.obj on Windows systems).

It performs the same optimizations as -ipo (Linux and Mac OS X) or /Qipo (Windows), but compilation stops before the final link stage, leaving an optimized object file that can be used in further link steps.

# **Alternate Options**

None

### See Also

ipo, Qipo compiler option

# ipo-jobs, Qipo-jobs

Specifies the number of commands (jobs) to be executed simultaneously during the link phase of Interprocedural Optimization (IPO).

### **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -ipo-jobsn

Windows: /Qipo-jobs:n

### Arguments

*n* Is the number of commands (jobs) to run

simultaneously. The number must be greater than

or equal to 1.

### Default

-ipo-jobs1 One command (job) is executed in an

or/Qipo-jobs:1 interprocedural optimization parallel build.

# Description

This option specifies the number of commands (jobs) to be executed simultaneously during the link phase of Interprocedural Optimization (IPO). It

should only be used if the link-time compilation is generating more than one object. In this case, each object is generated by a separate compilation, which can be done in parallel.

This option can be affected by the following compiler options:

- -ipo (Linux and Mac OS X) or /Qipo (Windows) when applications are large enough that the compiler decides to generate multiple object files.
- -ipon (Linux and Mac OS X) or /Qipon (Windows) when n is greater than 1.
- -ipo-separate (Linux) or /Qipo-separate (Windows)



Be careful when using this option. On a multi-processor system with lots of memory, it can speed application build time. However, if n is greater than the number of processors, or if there is not enough memory to avoid thrashing, this option can increase application build time.

# **Alternate Options**

None

#### See Also

ipo, Qipo compiler option

ipo-separate, Qipo-separate compiler option

# ipo-S, Qipo-S

Tells the compiler to optimize across multiple files and generate a single assembly file.

#### **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: -ipo-S Windows: /Qipo-S Arguments None Default OFF The compiler does not generate a multifile assembly file. Description This option tells the compiler to optimize across multiple files and generate a single assembly file (named ipo\_out.s on Linux and Mac OS X systems; ipo\_out.asm on Windows systems). It performs the same optimizations as -ipo (Linux and Mac OS X) or /Qipo (Windows), but compilation stops before the final link stage, leaving an optimized assembly file that can be used in further link steps. **Alternate Options** None See Also ipo, Qipo compiler option ipo-separate, Qipo-separate Tells the compiler to generate one object file for every source file. **IDE Equivalent** None

Architectures

IA-32, Intel® 64, IA-64 architectures

|--|

Linux: -ipo-separate

Mac OS X: None

Windows: /Qipo-separate

# Arguments

None

### Default

OFF The compiler decides whether to create one or more object files.

# Description

This option tells the compiler to generate one object file for every source file. It overrides any -ipo (Linux) or /Qipo (Windows) specification.

# **Alternate Options**

None

#### See Also

ipo, Qipo compiler option

# ivdep-parallel, Qivdep-parallel

Tells the compiler that there is no loop-carried memory dependency in the loop following an IVDEP directive.

# **IDE Equivalent**

Windows: Optimization > IVDEP Directive Memory Dependency

Linux: None

Mac OS X: None

#### Architectures

IA-64 architecture

**Syntax** 

Linux: -ivdep-parallel

Mac OS X: None

Windows: /Qivdep-parallel

Arguments

None

Default

OFF There may be loop-carried memory dependency in a loop that follows an IVDEP directive.

# Description

This option tells the compiler that there is no loop-carried memory dependency in the loop following an IVDEP There may be loop-carried memory dependency in a loop that follows an IVDEP directive.

This has the same effect as specifying the IVDEP:LOOP directive.

### **Alternate Options**

None

#### See Also

Optimizing Applications: Absence of Loop-carried Memory Dependency with IVDEP Directive

# fkeep-static-consts, Qkeep-static-consts

Tells the compiler to preserve allocation of variables that are not referenced in the source.

# IDE Equivalent

### None

### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -fkeep-static-consts

-fno-keep-static-consts

Windows: /Okeep-static-consts

/Qkeep-static-consts-

# Arguments

None

#### Default

-fno-keep- If a variable is never referenced in a routine, the static-consts or variable is discarded unless optimizations are /Qkeep-static- disabled by option -00 (Linux and Mac OS X) or

consts- /Od (Windows).

### Description

This option tells the compiler to preserve allocation of variables that are not referenced in the source.

The negated form can be useful when optimizations are enabled to reduce the memory usage of static data.

### **Alternate Options**

None

#### Qlocation

Specifies the directory for supporting tools.

# IDE Equivalent

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -Qlocation, string, dir

Windows: /Qlocation, string, dir

### Arguments

string Is the name of the tool.

dir Is the directory (path) where the tool is located.

#### Default

OFF The compiler looks for tools in a default area.

### Description

This option specifies the directory for supporting tools.

string can be any of the following:

- f Indicates the Intel Fortran compiler.
- fpp (or cpp) Indicates the Intel Fortran preprocessor.
- asm Indicates the assembler.
- link Indicates the linker.
- prof Indicates the profiler.
- On Windows systems, the following is also available:
- o masm Indicates the Microsoft assembler.
- On Linux and Mac OS X systems, the following are also available:
- o as Indicates the assembler.
- o gas Indicates the GNU assembler.
- Id Indicates the loader.

- o gld Indicates the GNU loader.
- o lib Indicates an additional library.
- crt Indicates the crt%.o files linked into executables to contain the place to start execution.

# **Alternate Options**

None

# Example

The following command provides the path for the fpp tool:

```
ifort -Qlocation,fpp,/usr/preproc myprog.f
```

#### See Also

<u>Qoption</u> compiler option

# lowercase, Qlowercase

See <u>names</u>.

# map-opts, Qmap-opts

Maps one or more compiler options to their equivalent on a different operating system.

# **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux: -map-opts

Mac OS X: None

Windows: /Qmap-opts

# Arguments

None

#### Default

OFF No platform mappings are performed.

# Description

This option maps one or more compiler options to their equivalent on a different operating system. The result is output to stdout.

On Windows systems, the options you provide are presumed to be Windows options, so the options that are output to stdout will be Linux equivalents.

On Linux systems, the options you provide are presumed to be Linux options, so the options that are output to stdout will be Windows equivalents.

The tool can be invoked from the compiler command line or it can be used directly.

No compilation is performed when the option mapping tool is used.

This option is useful if you have both compilers and want to convert scripts or makefiles.



Compiler options are mapped to their equivalent on the architecture you are using.

For example, if you are using a processor with IA-32 architecture, you will only see equivalent options that are available on processors with IA-32 architecture.

# **Alternate Options**

None

### Example

The following command line invokes the option mapping tool, which maps the Linux options to Windows-based options, and then outputs the results to stdout:

```
ifort -map-opts -xP -02
```

The following command line invokes the option mapping tool, which maps the Windows options to Linux-based options, and then outputs the results to stdout:

```
ifort /Qmap-opts /QxP /O2
```

#### See Also

Building Applications: Using the Option Mapping Tool

# no-bss-init, Qnobss-init

Tells the compiler to place in the DATA section any variables explicitly initialized with zeros.

# **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -no-bss-init

Windows: /Qnobss-init

# Arguments

None

#### Default

OFF Variables explicitly initialized with zeros are placed in the BSS section.

# Description

This option tells the compiler to place in the DATA section any variables explicitly initialized with zeros.

# **Alternate Options**

Linux and Mac OS X: -nobss-init (this is a deprecated option)

Windows: None

# onetrip, Qonetrip

Tells the compiler to follow the FORTRAN 66 Standard and execute DO loops at least once.

# **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -onetrip

Windows: /Qonetrip

# Arguments

None

### Default

OFF The compiler applies the current Fortran Standard semantics, which allows zero-trip DO loops.

# Description

This option tells the compiler to follow the FORTRAN 66 Standard and execute DO loops at least once.

# **Alternate Options**

Linux and Mac OS X: -1

Windows: /1

# openmp, Qopenmp

Enables the parallelizer to generate multi-threaded code based on the OpenMP\* directives.

# **IDE Equivalent**

Windows: Language > Process OpenMP Directives

Linux: None

Mac OS X: None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -openmp

Windows: /Qopenmp

# **Arguments**

None

### Default

OFF No OpenMP multi-threaded code is generated by the compiler.

# Description

This option enables the parallelizer to generate multi-threaded code based on the OpenMP\* directives. The code can be executed in parallel on both uniprocessor and multiprocessor systems.

If you use this option, multithreaded libraries are used, but option fpp is not automatically invoked.

This option sets option automatic.

This option works with any optimization level. Specifying no optimization (-00 on Linux or /od on Windows) helps to debug OpenMP applications.



On Mac OS X systems, when you enable OpenMP\*, you must also set the DYLD\_LIBRARY\_PATH environment variable within Xcode or an error will be displayed.

### **Alternate Options**

None

### See Also

openmp-stubs, Qopenmp-stubs compiler option

# openmp-lib, Qopenmp-lib

Lets you specify an OpenMP\* run-time library to use for linking.

# **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux: -openmp-lib type

Mac OS X: None

Windows: /Qopenmp-lib:type

### Arguments

type Specifies the type of library to use; it implies

compatibility levels. Possible values are:

legacy Tells the compiler to use the

legacy OpenMP\* run-time library (libguide). This setting does not provide compatibility with object files created using other compilers. This is a

<u>deprecated</u> option.

compat Tells the compiler to use the

compatibility OpenMP\* runtime library (libiomp). This setting provides compatibility with object files created using

Microsoft\* and GNU\*

compilers.

#### Default

-openmp-lib compat The compiler uses the compatibility or/Qopenmp-lib:compat OpenMP\* run-time library (libiomp).

### Description

This option lets you specify an OpenMP\* run-time library to use for linking.

The legacy OpenMP run-time library is not compatible with object files created using OpenMP run-time libraries supported in other compilers.

The compatibility OpenMP run-time library is compatible with object files created using the Microsoft\* OpenMP run-time library (vcomp) and GNU OpenMP run-time library (libgomp).

To use the compatibility OpenMP run-time library, compile and link your application using the <code>-openmp-lib</code> <code>compat</code> (Linux) or <code>/Qopenmp-lib:compat</code> (Windows) option. To use this option, you must also specify one of the following compiler options:

- Linux OS: -openmp, -openmp-profile, or -openmp-stubs
- Windows OS: /Qopenmp, /Qopenmp-profile, or /Qopenmp-stubs
  On Windows\* systems, the compatibility OpenMP\* run-time library lets you
  combine OpenMP\* object files compiled with the Microsoft\* C/C++ compiler with
  OpenMP\* object files compiled with the Intel C/C++ or Fortran compilers. The
  linking phase results in a single, coherent copy of the run-time library.
  On Linux\* systems, the compatibility Intel OpenMP\* run-time library lets you
  combine OpenMP\* object files compiled with the GNU\* gcc or gfortran compilers
  with similar OpenMP\* object files compiled with the Intel C/C++ or Fortran
  compilers. The linking phase results in a single, coherent copy of the run-time
  library.

You cannot link object files generated by the Intel® Fortran compiler to object files compiled by the GNU Fortran compiler, regardless of the presence or absence of the <code>-openmp</code> (Linux) or <code>/Qopenmp</code> (Windows) compiler option. This is because the Fortran run-time libraries are incompatible.



The compatibility OpenMP run-time library is not compatible with object files created using versions of the Intel compiler earlier than 10.0.

### **Alternate Options**

None

# See Also

openmp, Qopenmp compiler option
openmp-stubs, Qopenmp-stubs compiler option
openmp-profile, Qopenmp-profile compiler option

# openmp-link, Qopenmp-link

Controls whether the compiler links to static or dynamic OpenMP run-time libraries.

# **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -openmp-link library

Windows: /Qopenmp-link: library

# Arguments

library Specifies the OpenMP library to use. Possible

values are:

static Tells the compiler to link to

static OpenMP run-time

libraries.

dynamic Tells the compiler to link to

dynamic OpenMP run-time

libraries.

#### Default

-openmp-link dynamic The compiler links to dynamic OpenMP or /Qopenmp- run-time libraries. However, if option

link:dynamic

static is specified, the compiler links to

static OpenMP run-time libraries.

### Description

This option controls whether the compiler links to static or dynamic OpenMP runtime libraries.

To link to the static OpenMP run-time library (RTL) and create a purely static executable, you must specify <code>-openmp-link</code> static (Linux and Mac OS X) or <code>/Qopenmp-link:static</code> (Windows). However, we strongly recommend you use the default setting, <code>-openmp-link</code> dynamic (Linux and Mac OS X) or <code>/Qopenmp-link:dynamic</code> (Windows).



Compiler options -static-intel and -shared-intel (Linux and Mac OS X) have no effect on which OpenMP run-time library is linked.

# **Alternate Options**

None

# openmp-profile, Qopenmp-profile

Enables analysis of OpenMP\* applications if Intel® Thread Profiler is installed.

# IDE Equivalent

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux: -openmp-profile

Mac OS X: None

Windows:

/Qopenmp-profile

**Arguments** 

None

Default

OFF OpenMP applications are not analyzed.

### Description

This option enables analysis of OpenMP\* applications. To use this option, you must have previously installed Intel® Thread Profiler, which is one of the Intel® Threading Analysis Tools.

This option can adversely affect performance because of the additional profiling and error checking invoked to enable compatibility with the threading tools. Do not use this option unless you plan to use the Intel® Thread Profiler.

For more information about Intel® Thread Profiler (including an evaluation copy) open the page associated with threading tools at <a href="Intel® Software Development">Intel® Software Development</a> Products.

**Alternate Options** 

None

openmp-report, Qopenmp-report

Controls the OpenMP\* parallelizer's level of diagnostic messages.

**IDE Equivalent** 

Windows: Compilation Diagnostics > OpenMP Diagnostic Level

Linux: None

Mac OS X: Compiler Diagnostics > OpenMP Report

Architectures

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -openmp-report[n]

Windows: /Qopenmp-report[n]

# Arguments

*n* Is the level of diagnostic messages to display.

Possible values are:

0 No diagnostic messages are

displayed.

1 Diagnostic messages are

displayed indicating loops,

regions, and sections

successfully parallelized.

2 The same diagnostic

messages are displayed as

specified by openmp\_report1

plus diagnostic messages

indicating successful handling

of MASTER constructs,

SINGLE constructs, CRITICAL

constructs, ORDERED

constructs, ATOMIC directives,

and so forth.

### Default

-openmp-report 1 This is the default if you do not specify n.

or/Qopenmp-report1 The compiler displays diagnostic messages

indicating loops, regions, and sections successfully parallelized. If you do not specify the option on the command line, the default is to display no messages.

# Description

This option controls the OpenMP\* parallelizer's level of diagnostic messages. To use this option, you must also specify <code>-openmp</code> (Linux and Mac OS X) or <code>/Qopenmp</code> (Windows).

If this option is specified on the command line, the report is sent to stdout.

On Windows systems, if this option is specified from within the IDE, the report is included in the build log if the Generate Build Logs option is selected.

# **Alternate Options**

None

### See Also

<u>openmp</u>, <u>Qopenmp</u> compiler option Optimizing Applications:

Using Parallelism

OpenMP\* Report

# openmp-stubs, Qopenmp-stubs

Enables compilation of OpenMP programs in sequential mode.

### **IDE Equivalent**

Windows: Language > Process OpenMP Directives

Linux: None

Mac OS X: Language > Process OpenMP Directives

### **Architectures**

Intel® Fortran Compiler User and Reference Guides IA-32, Intel® 64, IA-64 architectures **Syntax** Linux and Mac OS X: -openmp-stubs Windows: /Qopenmp-stubs Arguments None Default OFF The library of OpenMP function stubs is not linked. Description This option enables compilation of OpenMP programs in sequential mode. The OpenMP directives are ignored and a stub OpenMP library is linked. **Alternate Options** None See Also openmp, Qopenmp compiler option openmp-threadprivate, Qopenmp-threadprivate Lets you specify an OpenMP\* threadprivate implementation. **IDE Equivalent** None

**Architectures** 

IA-32, Intel® 64, IA-64 architectures

Linux: -openmp-threadprivate type

Mac OS X: None

Windows: /Qopenmp-threadprivate:type

### **Arguments**

*type* Specifies the type of threadprivate implementation.

Possible values are:

legacy Tells the compiler to use the

legacy OpenMP\* threadprivate

implementation used in the

previous releases of the Intel®

compiler. This setting does not

provide compatibility with the implementation used by other

compilers.

compat Tells the compiler to use the

compatibility OpenMP\*

threadprivate implementation

based on applying the thread-

local attribute to each

threadprivate variable. This

setting provides compatibility

with the implementation

provided by the Microsoft\* and

GNU\* compilers.

#### Default

-openmp-threadprivate The compiler uses the legacy OpenMP\*
legacy threadprivate implementation used in the
or/Qopenmp- previous releases of the Intel® compiler.

threadprivate:legacy

### Description

This option lets you specify an OpenMP\* threadprivate implementation.

The legacy OpenMP run-time library is not compatible with object files created using OpenMP run-time libraries supported in other compilers.

To use this option, you must also specify one of the following compiler options:

- Linux OS: -openmp, -openmp-profile, or -openmp-stubs
- Windows OS: /Qopenmp, /Qopenmp-profile, or /Qopenmp-stubs

The value specified for this option is independent of the value used for option - openmp-lib (Linux) or /Qopenmp-lib (Windows).

# **Alternate Options**

None

# opt-block-factor, Qopt-block-factor

Lets you specify a loop blocking factor.

# **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

**Linux and Mac OS X**: -opt-block-factor=*n* 

Windows: /Qopt-block-factor:n

#### Arguments

*n* Is the blocking factor. It must be an integer. The

compiler may ignore the blocking factor if the value

is 0 or 1.

### Default

OFF The compiler uses default heuristics for loop blocking.

# Description

This option lets you specify a loop blocking factor.

# **Alternate Options**

None

# opt-jump-tables, Qopt-jump-tables

Enables or disables generation of jump tables for switch statements.

# **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -opt-jump-tables=keyword

-no-opt-jump-tables

Windows: /Qopt-jump-tables:keyword

/Qopt-jump-tables-

# Arguments

*keyword* Is the instruction for generating jump tables.

Possible values are:

never Tells the compiler to never

generate jump tables. All

switch statements are implemented as chains of if-then-elses. This is the same as specifying -no-opt-jump-tables (Linux and Mac OS) or /Qopt-jump-tables- (Windows).

default The compiler uses default

heuristics to determine when to

generate jump tables.

large Tells the compiler to generate

jump tables up to a certain pre-

defined size (64K entries).

n Must be an integer. Tells the

compiler to generate jump

tables up ton entries in size.

#### Default

-opt-jump-tables=default The compiler uses default heuristics or/Qopt-jump-tables:default to determine when to generate jump tables for switch statements.

# Description

This option enables or disables generation of jump tables for switch statements. When the option is enabled, it may improve performance for programs with large switch statements.

### **Alternate Options**

None

# opt-loadpair, Qopt-loadpair

Enables or disables loadpair optimization.

# **IDE Equivalent**

None

#### Architectures

IA-64 architecture

# Syntax

Linux: -opt-loadpair

-no-opt-loadpair

Mac OS X: None

Windows: /Qopt-loadpair

/Qopt-loadpair-

# Arguments

None

#### Default

-no-opt-loadpair Loadpair optimization is disabled

or/Qopt-loadpair- unless option 03 is specified.

# Description

This option enables or disables loadpair optimization.

When -03 is specified on IA-64 architecture, loadpair optimization is enabled by default. To disable loadpair generation, specify -no-opt-loadpair (Linux) or /Qopt-loadpair- (Windows).

# **Alternate Options**

None

# opt-mem-bandwidth, Qopt-mem-bandwidth

Enables performance tuning and heuristics that control memory bandwidth use among processors.

### **IDE Equivalent**

None

#### Architectures

IA-64 architecture

# Syntax

Linux: -opt-mem-bandwidthn

Mac OS X: None

Windows: /Qopt-mem-bandwidthn

# Arguments

*n* Is the level of optimizing for memory bandwidth

usage. Possible values are:

0 Enables a set of performance

tuning and heuristics in

compiler optimizations that is

optimal for serial code.

1 Enables a set of performance

tuning and heuristics in

compiler optimizations for

multithreaded code generated

by the compiler.

2 Enables a set of performance

tuning and heuristics in

compiler optimizations for

parallel code such as Windows
Threads, pthreads, and MPI
code, besides multithreaded
code generated by the
compiler.

#### Default

-opt-membandwidth0 or/Qopt-membandwidth0 For serial (non-parallel) compilation, a set of performance tuning and heuristics in compiler optimizations is enabled that is optimal for serial code.

-opt-membandwidth1 or/Qopt-membandwidth1

If you specify compiler option -parallel (Linux) or /Qparallel (Windows), -openmp (Linux) or /Qopenmp (Windows), or Cluster OpenMP option -cluster-openmp (Linux), a set of performance tuning and heuristics in compiler optimizations for multithreaded code generated by the compiler is enabled.

# Description

This option enables performance tuning and heuristics that control memory bandwidth use among processors. It allows the compiler to be less aggressive with optimizations that might consume more bandwidth, so that the bandwidth can be well-shared among multiple processors for a parallel program. For values of n greater than 0, the option tells the compiler to enable a set of performance tuning and heuristics in compiler optimizations such as prefetching, privatization, aggressive code motion, and so forth, for reducing memory bandwidth pressure and balancing memory bandwidth traffic among threads.

This option can improve performance for threaded or parallel applications on multiprocessors or multicore processors, especially when the applications are bounded by memory bandwidth.

# **Alternate Options**

None

#### See Also

parallel, Qparallel compiler option
openmp, Qopenmp compiler option

# opt-mod-versioning, Qopt-mod-versioning

Enables or disables versioning of modulo operations for certain types of operands.

# **IDE Equivalent**

None

#### Architectures

IA-64 architecture

# **Syntax**

Linux: -opt-mod-versioning

-no-opt-mod-versioning

Mac OS X: None

Windows: /Qopt-mod-versioning

/Qopt-mod-versioning-

# Arguments

None

# Default

### Description

This option enables or disables versioning of modulo operations for certain types of operands. It is used for optimization tuning.

Versioning of modulo operations may improve performance for x mod y when modulus y is a power of 2.

# **Alternate Options**

None

# opt-multi-version-aggressive, Qopt-multi-version-aggressive

Tells the compiler to use aggressive multi-versioning to check for pointer aliasing and scalar replacement.

### **IDE Equivalent**

None

### **Architectures**

IA-32, Intel® 64 architectures

### **Syntax**

Linux and Mac OS X: -opt-multi-version-aggressive

-no-opt-multi-version-aggressive

Windows: /Qopt-multi-version-aggressive

/Qopt-multi-version-aggressive-

### Arguments

None

### Default

-no-opt-multi-versionaggressive
Or/Qopt-multi-version-

The compiler uses default heuristics when checking for pointer aliasing and scalar replacement.

### Description

aggressive-

This option tells the compiler to use aggressive multi-versioning to check for pointer aliasing and scalar replacement. This option may improve performance.

### **Alternate Options**

None

# opt-prefetch, Qopt-prefetch

Enables or disables prefetch insertion optimization.

# **IDE Equivalent**

Windows: **Optimization > Prefetch Insertion** 

Linux: None

Mac OS X: Optimization > Enable Prefetch Insertion

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -opt-prefetch[=n]

-no-opt-prefetch

Windows: /Qopt-prefetch[:n]

/Qopt-prefetch-

# Arguments

*n* Is the level of detail in the report. Possible values

are:

O Disables software prefetching.

This is the same as specifying

-no-opt-prefetch (Linux

and Mac OS X) or /Qopt-

prefetch- (Windows).

1 to 4 Enables different levels of

software prefetching. If you do

not specify a value for n, the

default is 2 on IA-32 and Intel®

64 architecture; the default is 3

on IA-64 architecture. Use

lower values to reduce the

amount of prefetching.

### **Default**

IA-64 architecture: -opt-

prefetch

or/Qopt-prefetch

On IA-64 architecture, prefetch

insertion optimization is enabled.

IA-32 architecture and Intel® 64

architecture:

-no-opt-prefetch

or/Qopt-prefetch-

On IA-32 architecture and Intel® 64

architecture, prefetch insertion

optimization is disabled.

#### Description

This option enables or disables prefetch insertion optimization. The goal of prefetching is to reduce cache misses by providing hints to the processor about when data should be loaded into the cache.

On IA-64 architecture, this option is enabled by default if you specify option 01 or higher. To disable prefetching at these optimization levels, specify -no-opt-prefetch (Linux and Mac OS X) or /Qopt-prefetch- (Windows).

On IA-32 architecture and Intel® 64 architecture, this option enables prefetching when higher optimization levels are specified.

### **Alternate Options**

Linux and Mac OS X: -prefetch (this is a deprecated option)

Windows: /Qprefetch (this is a deprecated option)

# opt-prefetch-initial-values, Qopt-prefetch-initial-values

Enables or disables prefetches that are issued before a loop is entered.

### **IDE Equivalent**

None

#### **Architectures**

IA-64 architecture

# **Syntax**

Linux: -opt-prefetch-initial-values

-no-opt-prefetch-initial-values

Mac OS X: None

Windows: /Qopt-prefetch-initial-values

/Qopt-prefetch-initial-values-

### Arguments

None

#### Default

-opt-prefetch-initial- Prefetches are issued before a loop

values is entered.

or/Qopt-prefetch-initial-

values

# Description

This option enables or disables prefetches that are issued before a loop is entered. These prefetches target the initial iterations of the loop.

When -O1 or higher is specified on IA-64 architecture, prefetches are issued before a loop is entered. To disable these prefetches, specify -no-opt-prefetch-initial-values (Linux) or /Qopt-prefetch-initial-values- (Windows).

# **Alternate Options**

None

# opt-prefetch-issue-excl-hint, Qopt-prefetch-issue-excl-hint

Determines whether the compiler issues prefetches for stores with exclusive hint.

# **IDE Equivalent**

None

#### Architectures

IA-64 architecture

# Syntax

Linux: -opt-prefetch-issue-excl-hint

-no-opt-prefetch-issue-excl-hint

Mac OS X: None

Windows: /Qopt-prefetch-issue-excl-hint

/Qopt-prefetch-issue-excl-hint-

### Arguments

# None

#### Default

-no-opt-prefetch-issueexcl-hint or/Qopt-prefetch-issueexcl-hintThe compiler does not issue prefetches for stores with exclusive hint.

# Description

This option determines whether the compiler issues prefetches for stores with exclusive hint. If option <code>-opt-prefetch-issue-excl-hint</code> (Linux) or <code>/Qopt-prefetch-issue-excl-hint</code> (Windows) is specified, the prefetches will be issued if the compiler determines it is beneficial to do so.

When prefetches are issued for stores with exclusive-hint, the cache-line is in "exclusive-mode". This saves on cache-coherence traffic when other processors try to access the same cache-line. This feature can improve performance tuning.

### **Alternate Options**

None

# opt-prefetch-next-iteration, Qopt-prefetch-next-iteration

Enables or disables prefetches for a memory access in the next iteration of a loop.

### **IDE Equivalent**

None

#### **Architectures**

IA-64 architecture

# **Syntax**

Linux: -opt-prefetch-next-iteration

-no-opt-prefetch-next-iteration

Mac OS X: None

Windows: /Qopt-prefetch-next-iteration

/Qopt-prefetch-next-iteration-

### Arguments

None

#### Default

-opt-prefetch-next- Prefetches are issued for a memory

iteration access in the next iteration of a

or/Qopt-prefetch-next- loop.

iteration

### Description

This option enables or disables prefetches for a memory access in the next iteration of a loop. It is typically used in a pointer-chasing loop.

When -O1 or higher is specified on IA-64 architecture, prefetches are issued for a memory access in the next iteration of a loop. To disable these prefetches, specify -no-opt-prefetch-next-iteration (Linux) or /Qopt-prefetch-next-iteration- (Windows).

### **Alternate Options**

None

## opt-ra-region-strategy, Qopt-ra-region-strategy

Selects the method that the register allocator uses to partition each routine into regions.

#### **IDE Equivalent**

### None

### **Architectures**

IA-32, Intel® 64 architectures

## **Syntax**

Linux and Mac OS X: -opt-ra-region-strategy[=keyword]

Windows: /Qopt-ra-region-strategy[:keyword]

## Arguments

keyword Is the method used for partitioning. Possible

values are:

routine Creates a single region for

each routine.

block Partitions each routine into one

region per basic block.

trace Partitions each routine into one

region per trace.

region Partitions each routine into one

region per loop.

default The compiler determines which

method is used for partitioning.

### Default

-opt-ra-region- The compiler determines which

strategy=default method is used for partitioning. This

or/Qopt-ra-region- is also the default if keyword is not

strategy:default specified.

### Description

This option selects the method that the register allocator uses to partition each routine into regions.

When setting default is in effect, the compiler attempts to optimize the tradeoff between compile-time performance and generated code performance.

This option is only relevant when optimizations are enabled (01 or higher).

## **Alternate Options**

None

#### See Also

o compiler option

## opt-report, Qopt-report

Tells the compiler to generate an optimization report to stderr.

## **IDE Equivalent**

Windows: Diagnostics > Optimization Diagnostics Level

Linux: None

Mac OS X: None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -opt-report[n]

Windows: /Qopt-report[:n]

## Arguments

*n* Is the level of detail in the report. Possible values

are:

Tells the compiler to generate

no optimization report.

Tells the compiler to generate a report with the minimum level of detail.

Tells the compiler to generate a report with the medium level of detail.

Tells the compiler to generate a report with the maximum level of detail.

### Default

report 2 or with medium detail. If you do not specify the option on the command line, the compiler does not generate an report: 2 optimization report.

## Description

This option tells the compiler to generate an optimization report to stderr.

## **Alternate Options**

None

### See Also

<u>opt-report-file</u>, <u>Qopt-report-file</u> compiler option Optimizing Applications: Optimizer Report Generation

## opt-report-file, Qopt-report-file

Specifies the name for an optimization report.

ID	EΕ	aui	val	ent

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -opt-report-file=file

Windows: /Qopt-report-file:file

### Arguments

file Is the name for the optimization report.

#### Default

OFF No optimization report is generated.

### Description

This option specifies the name for an optimization report. If you use this option, you do not have to specify -opt-report (Linux and Mac OS X) or /Qopt-report (Windows).

## **Alternate Options**

None

#### See Also

opt-report, Qopt-report compiler option

Optimizing Applications: Optimizer Report Generation

## opt-report-help, Qopt-report-help

Displays the optimizer phases available for report generation.

## **IDE Equivalent**

None

### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -opt-report-help

Windows: /Qopt-report-help

## Arguments

None

## Default

OFF No optimization reports are generated.

## Description

This option displays the optimizer phases available for report generation using - opt-report-phase (Linux and Mac OS X) or /Qopt-report-phase (Windows). No compilation is performed.

## **Alternate Options**

None

### See Also

```
opt-report, Qopt-report compiler option
opt-report-phase, Qopt-report-phase compiler option
```

# opt-report-phase, Qopt-report-phase

Specifies an optimizer phase to use when optimization reports are generated.

## **IDE Equivalent**

None

### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -opt-report-phase=phase

Windows: /Qopt-report-phase:phase

## **Arguments**

phase Is the phase to generate reports for. Some of the

possible values are:

ipo The Interprocedural Optimizer

phase

hlo The High Level Optimizer

phase

hpo The High Performance

Optimizer phase

ilo The Intermediate Language

Scalar Optimizer phase

ecg The Code Generator phase

(Windows and Linux systems

using IA-64 architecture only)

ecg\_swp The software pipelining

component of the Code

Generator phase (Windows

and Linux systems using IA-64

architecture only)

pgo The Profile Guided

Optimization phase

all All optimizer phases

Default

OFF No optimization reports are generated.

## Description

This option specifies an optimizer phase to use when optimization reports are generated. To use this option, you must also specify <code>-opt-report</code> (Linux and Mac OS X) or <code>/Qopt-report</code> (Windows).

This option can be used multiple times on the same command line to generate reports for multiple optimizer phases.

When one of the logical names for optimizer phases is specified for phase, all reports from that optimizer phase are generated.

To find all phase possibilities, use option -opt-report-help (Linux and Mac OS X) or /Qopt-report-help (Windows).

### **Alternate Options**

None

#### See Also

opt-report, Qopt-report compiler option

## opt-report-routine, Qopt-report-routine

Tells the compiler to generate reports on the routines containing specified text.

### **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -opt-report-routine=string Windows: /Qopt-report-routine:string Arguments Is the text (string) to look for. string Default OFF No optimization reports are generated. Description This option tells the compiler to generate reports on the routines containing specified text as part of their name. **Alternate Options** None See Also opt-report, Qopt-report compiler option opt-streaming-stores, Qopt-streaming-stores Enables generation of streaming stores for optimization.

## **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64 architectures

# **Syntax**

Linux and Mac OS X: -opt-streaming-stores *keyword*Windows: /Qopt-streaming-stores:*keyword* 

## Arguments

*keyword* Specifies whether streaming stores are generated.

Possible values are:

always Enables generation of

streaming stores for

optimization. The compiler

optimizes under the

assumption that the application

is memory bound.

never Disables generation of

streaming stores for

optimization. Normal stores are

performed.

auto Lets the compiler decide which

instructions to use.

#### Default

-opt- The compiler decides whether to use streaming stores or

streaming- normal stores.

stores auto

or/Qopt-

streaming-

stores:auto

### Description

This option enables generation of streaming stores for optimization. This method stores data with instructions that use a non-temporal buffer, which minimizes memory hierarchy pollution.

For this option to be effective, the compiler must be able to generate SSE2 (or higher) instructions. For more information, see compiler option x or ax.

This option may be useful for applications that can benefit from streaming stores.

## **Alternate Options**

None

#### See Also

```
ax, Qax compiler option
```

x, Qx compiler option

opt-mem-bandwidth, Qopt-mem-bandwidth, Qx compiler option

Optimizing Applications: Vectorization Support

# opt-subscript-in-range, Qopt-subscript-in-range

Determines whether the compiler assumes no overflows in the intermediate computation of subscript expressions in loops.

### **IDE Equivalent**

None

### **Architectures**

IA-32, Intel® 64 architectures

## **Syntax**

Linux and Mac OS X: -opt-subscript-in-range

-no-opt-subscript-in-range

Windows: /Qopt-subscript-in-range

/Qopt-subscript-in-range-

## Arguments

None

#### Default

-no-opt- The compiler assumes overflows in the

```
subscript-in- intermediate computation of subscript expressions
range in loops.

or/Qopt-
subscript-in-
range-
```

### Description

This option determines whether the compiler assumes no overflows in the intermediate computation of subscript expressions in loops.

If you specify -opt-subscript-in-range (Linux and Mac OS X) or /Qopt-subscript-in-range (Windows), the compiler ignores any data type conversions used and it assumes no overflows in the intermediate computation of subscript expressions. This feature can enable more loop transformations.

### **Alternate Options**

None

#### Example

The following shows an example where these options can be useful. m is declared as type integer(kind=8) (64-bits) and all other variables inside the subscript are declared as type integer(kind=4) (32-bits):

```
A[i + j + (n + k) * m]
```

## Qoption

Passes options to a specified tool.

## **IDE Equivalent**

None

## **Architectures**

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -Qoption, string, options

Windows: /Qoption, string, options

### Arguments

string Is the name of the tool.

options Are one or more comma-separated, valid options

for the designated tool.

#### Default

OFF No options are passed to tools.

## Description

This option passes options to a specified tool.

If an argument contains a space or tab character, you must enclose the entire argument in quotation marks (" "). You must separate multiple arguments with commas.

string can be any of the following:

- fpp (or cpp) Indicates the Intel Fortran preprocessor.
- asm Indicates the assembler.
- link Indicates the linker.
- prof Indicates the profiler.
- On Windows systems, the following is also available:
- o masm Indicates the Microsoft assembler.
- On Linux and Mac OS X systems, the following are also available:
- o as Indicates the assembler.
- gas Indicates the GNU assembler.
- o Id Indicates the loader.
- gld Indicates the GNU loader.
- lib Indicates an additional library.

 crt - Indicates the crt%.o files linked into executables to contain the place to start execution.

## **Alternate Options**

None

### Example

On Linux and Mac OS X systems:

The following example directs the linker to link with an alternative library:

```
ifort -Qoption, link, -L., -Lmylib prog1.f
```

The following example passes a compiler option to the assembler to generate a listing file:

```
ifort -Qoption,as,"-as=myprogram.lst" -use-asm myprogram.f90
On Windows systems:
```

The following example directs the linker to create a memory map when the compiler produces the executable file from the source being compiled:

```
ifort /Qoption,link,/map:progl.map progl.f
The following example passes a compiler option to the assembler:
```

```
ifort /Quse_asm /Qoption,masm,"/WX" myprogram.f90
```

#### See Also

Qlocation compiler option

qp

See p.

## pad, Qpad

Enables the changing of the variable and array memory layout.

## **IDE Equivalent**

None

### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -pad

-nopad

Windows: /Qpad

/Qpad-

## Arguments

None

#### Default

-nopad Variable and array memory layout is performed by defaultor methods.

/Qpad-

# Description

This option enables the changing of the variable and array memory layout.

This option is effectively not different from the align option when applied to structures and derived types. However, the scope of pad is greater because it applies also to common blocks, derived types, sequence types, and structures.

## **Alternate Options**

None

### See Also

align compiler option

## pad-source, Qpad-source

Specifies padding for fixed-form source records.

## **IDE Equivalent**

Windows: Language > Pad Fixed Form Source Lines

Linux: None

Mac OS X: Language > Pad Fixed Form Source Lines

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -pad-source

-nopad-source

Windows: /pad-source

/nopad-source

/Qpad-source

/Qpad-source-

## Arguments

None

## Default

-nopad- Fixed-form source records are not padded.

source

or

/Qpad-

source-

## Description

This option specifies padding for fixed-form source records. It tells the compiler that fixed-form source lines shorter than the statement field width are to be padded with spaces to the end of the statement field. This affects the interpretation of character and Hollerith literals that are continued across source records.

The default value setting causes a warning message to be displayed if a character or Hollerith literal that ends before the statement field ends is continued onto the next source record. To suppress this warning message, specify option – warn nousage (Linux and Mac OS X) or /warn:nousage (Windows).

Specifying pad-source or /Qpad-source can prevent warning messages associated with option -warn usage (Linux and Mac OS X) or /warn:usage (Windows).

### **Alternate Options**

None

#### See Also

warn compiler option

## **Qpar-adjust-stack**

Tells the compiler to generate code to adjust the stack size for a fiber-based main thread.

## **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64 architectures

## **Syntax**

Linux and Mac OS X: None

Windows: /Qpar-adjust-stack:n

### Arguments

*n* Is the stack size (in bytes) for the fiber-based main

thread. It must be a number equal to or greater

than zero.

#### Default

/Qpar-adjust-stack: 0 No adjustment is made to the main thread stack size.

## Description

This option tells the compiler to generate code to adjust the stack size for a fiberbased main thread. This can reduce the stack size of threads.

For this option to be effective, you must also specify option /Qparallel.

## **Alternate Options**

None

#### See Also

parallel, Qparallel compiler option

## par-report, Qpar-report

Controls the diagnostic information reported by the auto-parallelizer.

## IDE Equivalent

Windows: Compilation Diagnostics > Auto-Parallelizer Diagnostic Level

Linux: None

Mac OS X: Diagnostics > Auto-Parallelizer Report

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -par-report[n]

Windows: /Qpar-report[n]

### Arguments

Is a value denoting which diagnostic messages to report. Possible values are:

- Tells the auto-parallelizer to report no diagnostic information.
- Tells the auto-parallelizer to report diagnostic messages for loops successfully auto-parallelized. The compiler also issues a "LOOP AUTO-PARALLELIZED" message for parallel loops.
- Tells the auto-parallelizer to report diagnostic messages for loops successfully and unsuccessfully auto-parallelized.
- Tells the auto-parallelizer to report the same diagnostic messages specified by 2 plus additional information about any proven or assumed dependencies inhibiting autoparallelization (reasons for not parallelizing).

## Default

-par- If you do not specify *n*, the compiler displays diagnostic

report1 messages for loops successfully auto-parallelized. If you or/Qpar- do not specify the option on the command line, the default

report1 is to display no parallel disgnostic messages.

## Description

This option controls the diagnostic information reported by the auto-parallelizer (parallel optimizer). To use this option, you must also specify -parallel (Linux and Mac OS X) or /Qparallel (Windows).

If this option is specified on the command line, the report is sent to stdout.

On Windows systems, if this option is specified from within the IDE, the report is included in the build log if the Generate Build Logs option is selected.

#### **Alternate Options**

None

# par-runtime-control, Qpar-runtime-control

Generates code to perform run-time checks for loops that have symbolic loop bounds.

### **IDE Equivalent**

None

## Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -par-runtime-control

-no-par-runtime-control

Windows: /Qpar-runtime-control

/Qpar-runtime-control-

### Arguments

### None

#### Default

```
-no-par- The compiler uses default heuristics when runtime-control checking loops.

or/Qpar-runtime-
control-
```

### Description

This option generates code to perform run-time checks for loops that have symbolic loop bounds.

If the granularity of a loop is greater than the parallelization threshold, the loop will be executed in parallel.

If you do not specify this option, the compiler may not parallelize loops with symbolic loop bounds if the compile-time granularity estimation of a loop can not ensure it is beneficial to parallelize the loop.

# **Alternate Options**

None

## par-schedule, Qpar-schedule

Lets you specify a scheduling algorithm or a tuning method for loop iterations.

## **IDE Equivalent**

None

### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

**Linux and Mac OS X**: -par-schedule-*keyword*[=*n*]

Windows: /Qpar-schedule-keyword[[:]n]

## Arguments

keyword Specifies the scheduling algorithm or tuning

method. Possible values are:

auto Lets the compiler or run-time

system determine the scheduling algorithm.

static Divides iterations into

contiguous pieces.

static- Divides iterations into even-

balanced sized chunks.

static- Divides iterations into even-

steal sized chunks, but allows

threads to steal parts of chunks from neighboring

threads.

dynamic Gets a set of iterations

dynamically.

guided Specifies a minimum number

of iterations.

guided- Divides iterations by using

analytical exponential distribution or

dynamic distribution.

runtime Defers the scheduling decision

until run time.

Is the size of the chunk or the number of iterations

for each chunk. This setting can only be specified

for static, dynamic, and guided. For more information, see the descriptions of each keyword below.

## Default

static- Iterations are divided into even-sized chunks and the chunks balanced are assigned to the threads in the team in a round-robin fashion in the order of the thread number.

## Description

This option lets you specify a scheduling algorithm or a tuning method for loop iterations. It specifies how iterations are to be divided among the threads of the team.

This option affects performance tuning and can provide better performance during auto-parallelization.

Option	Description
-par-schedule-auto <b>or</b> /Qpar-	Lets the compiler or run-time system
schedule-auto	determine the scheduling algorithm. Any
	possible mapping may occur for
	iterations to threads in the team.
-par-schedule-static <b>or</b> /Qpar-	Divides iterations into contiguous pieces
schedule-static	(chunks) of size $n$ . The chunks are
	assigned to threads in the team in a
	round-robin fashion in the order of the
	thread number. Note that the last chunk
	to be assigned may have a smaller
	number of iterations.
	If no $n$ is specified, the iteration space is
	divided into chunks that are

Option	Description
	approximately equal in size, and each thread is assigned at most one chunk.
<pre>-par-schedule-static-balanced Or /Qpar-schedule-static- balanced</pre>	Divides iterations into even-sized chunks. The chunks are assigned to the threads in the team in a round-robin fashion in the order of the thread number.
-par-schedule-static-steal or /Qpar-schedule-static-steal	Divides iterations into even-sized chunks, but when a thread completes its chunk, it can steal parts of chunks assigned to neighboring threads.  Each thread keeps track of L and U, which represent the lower and upper bounds of its chunks respectively.  Iterations are executed starting from the lower bound, and simultaneously, L is updated to represent the new lower bound.
-par-schedule-dynamic or /Qpar-schedule-dynamic	Can be used to get a set of iterations dynamically. Assigns iterations to threads in chunks as the threads request them. The thread executes the chunk of iterations, then requests another chunk, until no chunks remain to be assigned.  As each thread finishes a piece of the iteration space, it dynamically gets the next set of iterations. Each chunk

Option	Description
	contains $n$ iterations, except for the last chunk to be assigned, which may have fewer iterations. If no $n$ is specified, the default is 1.
-par-schedule-guided Of /Qpar-schedule-guided	Can be used to specify a minimum number of iterations. Assigns iterations to threads in chunks as the threads request them. The thread executes the chunk of iterations, then requests another chunk, until no chunks remain to be assigned.  For a chunk of size 1, the size of each chunk is proportional to the number of unassigned iterations divided by the number of threads, decreasing to 1.  For an $n$ with value $k$ (greater than 1), the size of each chunk is determined in the same way with the restriction that the chunks do not contain fewer than $k$ iterations (except for the last chunk to be assigned, which may have fewer than $k$ iterations). If no $k$ is specified, the default is 1.
-par-schedule-guided- analytical or /Qpar-schedule- guided-analytical	Divides iterations by using exponential distribution or dynamic distribution. The method depends on run-time implementation. Loop bounds are calculated with faster synchronization and chunks are dynamically dispatched

Option	Description
	at run time by threads in the team.
-par-schedule-runtime or /Qpar-	Defers the scheduling decision until run
schedule-runtime	time. The scheduling algorithm and
	chunk size are then taken from the
	setting of environment variable
	OMP_SCHEDULE.

## **Alternate Options**

None

## par-threshold, Qpar-threshold

Sets a threshold for the auto-parallelization of loops.

## **IDE Equivalent**

Windows: **Optimization > Threshold For Auto-Parallelization** 

Linux: None

Mac OS X: Optimization > Threshold For Auto-Parallelization

## Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -par-threshold[n]

Windows: /Qpar-threshold[[:]n]

## Arguments

*n* Is an integer whose value is the threshold for the

auto-parallelization of loops. Possible values are 0

through 100.

If *n* is 0, loops get auto-parallelized always, regardless of computation work volume.

If *n* is 100, loops get auto-parallelized when performance gains are predicted based on the compiler analysis data. Loops get auto-parallelized only if profitable parallel execution is almost certain.

The intermediate 1 to 99 values represent the percentage probability for profitable speed-up. For example, n=50 directs the compiler to parallelize only if there is a 50% probability of the code speeding up if executed in parallel.

#### Default

-par- Loops get auto-parallelized only if profitable

threshold100 parallel execution is almost certain. This is also the

or/Qpar- default if you do not specify *n*.

threshold100

#### Description

This option sets a threshold for the auto-parallelization of loops based on the probability of profitable execution of the loop in parallel. To use this option, you must also specify <code>-parallel</code> (Linux and Mac OS X) or <code>/Qparallel</code> (Windows). This option is useful for loops whose computation work volume cannot be determined at compile-time. The threshold is usually relevant when the loop trip count is unknown at compile-time.

The compiler applies a heuristic that tries to balance the overhead of creating multiple threads versus the amount of work available to be shared amongst the threads.

### **Alternate Options**

None

# parallel, Qparallel

Tells the auto-parallelizer to generate multithreaded code for loops that can be safely executed in parallel.

### **IDE Equivalent**

Windows: **Optimization > Parallelization** 

Linux: None

Mac OS X: Optimization > Parallelization

### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -parallel

Windows: /Qparallel

## Arguments

None

#### Default

OFF Multithreaded code is not generated for loops that can be safely executed in parallel.

## Description

This option tells the auto-parallelizer to generate multithreaded code for loops that can be safely executed in parallel.

To use this option, you must also specify option 02 or 03.



On Mac OS X systems, when you enable automatic parallelization, you must also set the DYLD\_LIBRARY\_PATH environment variable within Xcode or an error will be displayed.

## **Alternate Options**

None

## See Also

o compiler option

## pc, Qpc

Enables control of floating-point significand precision.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64 architectures

## **Syntax**

Linux and Mac OS X: -pcn

Windows: /Qpcn

## Arguments

*n* Is the floating-point significand precision. Possible

values are:

Rounds the significand to 24

bits (single precision).

Rounds the significand to 53

bits (double precision).

80 Rounds the significand to 64

bits (extended precision).

Default

-pc80 On Linux\* and Mac OS\* X systems, the floating-point

or/Opc64 significand is rounded to 64 bits. On Windows\* systems, the

floating-point significand is rounded to 53 bits.

Description

This option enables control of floating-point significand precision.

Some floating-point algorithms are sensitive to the accuracy of the significand, or fractional part of the floating-point value. For example, iterative operations like

division and finding the square root can run faster if you lower the precision with

the this option.

Note that a change of the default precision control or rounding mode, for example,

by using the -pc32 (Linux and Mac OS X) or /Qpc32 (Windows) option or by

user intervention, may affect the results returned by some of the mathematical

functions.

**Alternate Options** 

None

See Also

Floating-point Operations: Floating-point Options Quick Reference

mp1, Qprec

Improves floating-point precision and consistency.

**IDE Equivalent** 

None

**Architectures** 

564

IA-32, Intel® 64, IA-64 architectures

### Syntax

Linux and Mac OS X: -mp1

Windows: /Qprec

### Arguments

None

#### Default

OFF The compiler provides good accuracy and run-time performance at the expense of less consistent floating-point results.

### Description

This option improves floating-point consistency. It ensures the out-of-range check of operands of transcendental functions and improves the accuracy of floating-point compares.

This option prevents the compiler from performing optimizations that change NaN comparison semantics and causes all values to be truncated to declared precision before they are used in comparisons. It also causes the compiler to use library routines that give better precision results compared to the X87 transcendental instructions.

This option disables fewer optimizations and has less impact on performance than option fltconsistency or mp.

### **Alternate Options**

None

#### See Also

<u>fltconsistency</u> compiler option

mp compiler option

# prec-div, Qprec-div

Improves precision of floating-point divides.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -prec-div

-no-prec-div

Windows: /Qprec-div

/Qprec-div-

## Arguments

None

### Default

```
-prec-div The compiler uses this method for floating-point divides.
```

or/Qprec-

div

## Description

This option improves precision of floating-point divides. It has a slight impact on speed.

With some optimizations, such as -xSSE2 (Linux) or /QxSSE2 (Windows), the compiler may change floating-point division computations into multiplication by the reciprocal of the denominator. For example, A/B is computed as A \* (1/B) to improve the speed of the computation.

However, sometimes the value produced by this transformation is not as accurate as full IEEE division. When it is important to have fully precise IEEE division, use this option to disable the floating-point division-to-multiplication optimization. The result is more accurate, with some loss of performance. If you specify -no-prec-div (Linux and Mac OS X) or /Qprec-div- (Windows), it enables optimizations that give slightly less precise results than full IEEE division.

## **Alternate Options**

None

### See Also

Floating-point Operations: Floating-point Options Quick Reference

## prec-sqrt, Qprec-sqrt

Improves precision of square root implementations.

### **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64 architectures

### **Syntax**

Linux and Mac OS X: -prec-sqrt

-no-prec-sqrt

Windows: /Qprec-sqrt

/Qprec-sqrt-

### Arguments

None

## Default

-no-prec- The compiler uses a faster but less precise

sqrt implementation of square root.

or /Qprec- Note that the default is -prec-sqrt or /Qprec-sqrt if

sqrt- any of the following options are specified: /Od, /Op, or

/Qprec on Windows systems; -00, -mp (or -

fltconsistency), or -mp1 on Linux and Mac OS X

systems.

## Description

This option improves precision of square root implementations. It has a slight impact on speed.

This option inhibits any optimizations that can adversely affect the precision of a square root computation. The result is fully precise square root implementations, with some loss of performance.

### **Alternate Options**

None

## prof-data-order, Qprof-data-order

Enables or disables data ordering if profiling information is enabled.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux: -prof-data-order

-no-prof-data-order

Mac OS X: None

Windows: /Qprof-data-order

/Qprof-data-order-

### Arguments

None

#### Default

```
-no-prof- Data ordering is disabled.

data-order

or/Qprof-
data-
order-
```

### Description

This option enables or disables data ordering if profiling information is enabled. It controls the use of profiling information to order static program data items.

For this option to be effective, you must do the following:

- For instrumentation compilation, you must specify -prof-gen=globdata (Linux) or /Qprof-gen:globdata (Windows).
- For feedback compilation, you must specify <code>-prof-use</code> (Linux) or <code>/Qprof-use</code> (Windows). You must not use multi-file optimization by specifying options such as option <code>-ipo</code> (Linux) or <code>/Qipo</code> (Windows), or option <code>-ipo-c</code> (Linux) or <code>/Qipo-c</code> (Windows).

### **Alternate Options**

None

#### See Also

```
<u>prof-gen</u>, <u>Qprof-gen</u> compiler option <u>prof-use</u>, <u>Qprof-use</u> compiler option
```

prof-func-order, Qprof-func-order compiler option

## prof-dir, Qprof-dir

Specifies a directory for profiling information output files.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -prof-dir dir

Windows: /Qprof-dir dir

## Arguments

*dir* Is the name of the directory.

## Default

OFF Profiling output files are placed in the directory where the program is compiled.

## Description

This option specifies a directory for profiling information output files (\*.dyn and \*.dpi). The specified directory must already exist.

You should specify this option using the same directory name for both instrumentation and feedback compilations. If you move the .dyn files, you need to specify the new path.

## **Alternate Options**

None

### See Also

Floating-point Operations:

Profile-guided Optimization (PGO) Quick Reference

Coding Guidelines for Intel(R) Architectures

# prof-file, Qprof-file

Specifies an alternate file name for the profiling summary files.

# IDE Equivalent

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -prof-file file

Windows: /Qprof-file file

### **Arguments**

file Is the name of the profiling summary file.

#### Default

OFF The profiling summary files have the file name pgopti.\*

# Description

This option specifies an alternate file name for the profiling summary files. The *file* is used as the base name for files created by different profiling passes. If you add this option to profmerge, the .dpi file will be named *file*.dpi instead of pgopti.dpi.

If you specify <code>-prof-genx</code> (Linux and Mac OS X) or <code>/Qprof-genx</code> (Windows) with this option, the .spi and .spl files will be named <code>file</code>.spi and <code>file</code>.spl instead of pgopti.spi and pgopti.spl.

If you specify -prof-use (Linux and Mac OS X) or /Qprof-use (Windows) with this option, the .dpi file will be named *file*.dpi instead of pgopti.dpi.

## **Alternate Options**

None

#### See Also

prof-gen, Qprof-gen compiler option
prof-use, Qprof-use compiler option

Optimizing Applications:

Profile-guided Optimizations Overview

Coding Guidelines for Intel(R) Architectures

Profile an Application

# prof-func-order, Qprof-func-order

Enables or disables function ordering if profiling information is enabled.

## IDE Equivalent

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux: -prof-func-order

-no-prof-func-order

Mac OS X: None

Windows: /Qprof-func-order

/Qprof-func-order-

### Arguments

None

#### Default

```
-no-prof- Function ordering is disabled.

func-order

or/Qprof-

func-

order-
```

### Description

This option enables or disables function ordering if profiling information is enabled.

For this option to be effective, you must do the following:

- For instrumentation compilation, you must specify -prof-gen=srcpos (Linux) or /Qprof-gen:srcpos (Windows).
- For feedback compilation, you must specify -prof-use (Linux) or /Qprof-use (Windows). You must not use multi-file optimization by specifying options such as option -ipo (Linux) or /Qipo (Windows), or option -ipo-c (Linux) or /Qipo-c (Windows).

If you enable profiling information by specifying option <code>-prof-use</code> (Linux) or <code>/Qprof-use</code> (Windows), <code>-prof-func-groups</code> (Linux) and <code>/Qprof-func-groups</code> (Windows) are set and function grouping is enabled. However, if you explicitly enable <code>-prof-func-order</code> (Linux) or <code>/Qprof-func-order</code> (Windows), function ordering is performed instead of function grouping. On Linux\* systems, this option is only available for Linux linker 2.15.94.0.1, or later.

To set the hotness threshold for function grouping and function ordering, use option -prof-hotness-threshold (Linux) or /Qprof-hotness-threshold (Windows).

## **Alternate Options**

### None

The following example shows how to use this option on a Windows system:

```
ifort /Qprof-gen:globdata file1.f90 file2.f90 /exe:instrumented.exe
    ./instrumented.exe
ifort /Qprof-use /Qprof-func-order file1.f90 file2.f90
/exe:feedback.exe
```

The following example shows how to use this option on a Linux system:

#### See Also

```
prof-hotness-threshold, Qprof-hotness-threshold compiler option
prof-gen, Qprof-gen compiler option
prof-use, Qprof-use compiler option
prof-data-order, Qprof-data-order compiler option
prof-func-groups compiler option
```

## prof-gen, Qprof-gen

Produces an instrumented object file that can be used in profile-guided optimization.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

```
Linux and Mac OS X: -prof-gen[=keyword]
-no-prof-gen

Windows: /Qprof-gen[:keyword]
/Oprof-gen-
```

#### **Arguments**

keyword

Specifies details for the instrumented file. Possible values are:

default

Produces an instrumented object file. This is the same as specifying -prof-gen (Linux\* and Mac OS\* X) or /Qprof-gen (Windows\*) with no

keyword.

srcpos

Produces an instrumented object file that includes extra source position information. This option is the same as option -prof-genx (Linux\* and Mac OS\* X) or /Qprof-genx (Windows\*), which are

deprecated.

globdata

Produces an instrumented object file that includes information for global data

layout.

#### Default

-no- Profile generation is disabled.

prof-

gen **or** 

/Qprof-

gen-

## Description

This option produces an instrumented object file that can be used in profile-

guided optimization. It gets the execution count of each basic block.

If you specify keyword srcpos or globdata, a static profile information file (.spi)

is created. These settings may increase the time needed to do a parallel build

using -prof-gen, because of contention writing the .spi file.

These options are used in phase 1 of the Profile Guided Optimizer (PGO) to

instruct the compiler to produce instrumented code in your object files in

preparation for instrumented execution.

## **Alternate Options**

None

### See Also

Optimizing Applications:

Basic PGO Options

Example of Profile-Guided Optimization

# prof-genx, Qprof-genx

This is a deprecated option. See <a href="prof-gen">prof-gen</a> keyword <a href="srcpos.">srcpos</a>.

# prof-hotness-threshold, Qprof-hotness-threshold

Lets you set the hotness threshold for function grouping and function ordering.

### **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

### Syntax

Linux: -prof-hotness-threshold=n

Mac OS X: None

Windows: /Qprof-hotness-threshold:n

#### Arguments

*n* Is the hotness threshold. *n* is a percentage having

a value between 0 and 100 inclusive. If you specify

0, there will be no hotness threshold setting in

effect for function grouping and function ordering.

## Default

OFF The compiler's default hotness threshold setting of 10

percent is in effect for function grouping and function

ordering.

### Description

This option lets you set the hotness threshold for function grouping and function ordering.

The "hotness threshold" is the percentage of functions in the application that should be placed in the application's hot region. The hot region is the most frequently executed part of the application. By grouping these functions together into one hot region, they have a greater probability of remaining resident in the instruction cache. This can enhance the application's performance.

For this option to take effect, you must specify option <code>-prof-use</code> (Linux) or <code>/Qprof-use</code> (Windows) and one of the following:

- On Linux systems: -prof-func-groups or -prof-func-order
- On Windows systems: /Qprof-func-order

# **Alternate Options**

None

## See Also

```
<u>prof-use</u>, <u>Qprof-use</u> compiler option<u>prof-func-groups</u> compiler option<u>prof-func-order</u>, <u>Qprof-func-order</u> compiler option
```

## prof-src-dir, Qprof-src-dir

Determines whether directory information of the source file under compilation is considered when looking up profile data records.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

```
Linux and Mac OS X: -prof-src-dir
```

-no-prof-src-dir

Windows: /Qprof-src-dir

/Qprof-src-dir-

### **Arguments**

None

### **Default**

```
-prof- Directory information is used when looking up profile data src-dir records in the .dpi file.

or/Qprof-
src-dir
```

# Description

This option determines whether directory information of the source file under compilation is considered when looking up profile data records in the .dpi file. To

use this option, you must also specify option -prof-use (Linux and Mac OS X) or /Qprof-use (Windows).

If the option is enabled, directory information is considered when looking up the profile data records within the .dpi file. You can specify directory information by using one of the following options:

- Linux and Mac OS X: -prof-src-root or -prof-src-root-cwd
- Windows: /Qprof-src-root or /Qprof-src-root-cwd

If the option is disabled, directory information is ignored and only the name of the file is used to find the profile data record.

Note that options <code>-prof-src-dir</code> (Linux and Mac OS X) and <code>/Qprof-src-dir</code> (Windows) control how the names of the user's source files get represented within the .dyn or .dpi files. Options <code>-prof-dir</code> (Linux and Mac OS X) and <code>/Qprof-dir</code> (Windows) specify the location of the .dyn or the .dpi files.

### **Alternate Options**

None

### See Also

```
prof-use, Qprof-use compiler option
prof-src-root, Qprof-src-root compiler option
prof-src-root-cwd, Qprof-src-root-cwd compiler option
```

## prof-src-root, Qprof-src-root

Lets you use relative directory paths when looking up profile data and specifies a directory as the base.

### **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -prof-src-root=dir

Windows: /Qprof-src-root:dir

### Arguments

dir Is the base for the relative paths.

#### Default

OFF The setting of relevant options determines the path used when looking up profile data records.

## Description

This option lets you use relative directory paths when looking up profile data in .dpi files. It lets you specify a directory as the base. The paths are relative to a base directory specified during the <code>-prof-gen</code> (Linux and Mac OS X) or <code>/Oprof-gen</code> (Windows) compilation phase.

This option is available during the following phases of compilation:

- Linux and Mac OS X: -prof-gen and -prof-use phases
- Windows: /Qprof-gen and /Qprof-use phases

When this option is specified during the <code>-prof-gen</code> or <code>/Qprof-gen</code> phase, it stores information into the .dyn or .dpi file. Then, when .dyn files are merged together or the .dpi file is loaded, only the directory information below the root directory is used for forming the lookup key.

When this option is specified during the <code>-prof-use</code> or <code>/Qprof-use</code> phase, it specifies a root directory that replaces the root directory specified at the <code>-prof-gen</code> or <code>/Qprof-gen</code> phase for forming the lookup keys.

To be effective, this option or option -prof-src-root-cwd (Linux and Mac OS X) or /Qprof-src-root-cwd (Windows) must be specified during the -prof-gen or /Qprof-gen phase. In addition, if one of these options is not specified, absolute paths are used in the .dpi file.

### **Alternate Options**

### None

Consider the initial -prof-gen compilation of the source file c:\user1\feature\_foo\myproject\common\glob.f90:

```
ifort -prof-gen -prof-src-root=c:\user1\feature_foo\myproject -c
common\glob.f90
```

For the -prof-use phase, the file glob.f90 could be moved into the directory c:\user2\feature\_bar\myproject\common\glob.f90 and profile information would be found from the .dpi when using the following:

```
ifort -prof-use -prof-src-root=c:\user2\feature_bar\myproject -c
common\glob.f90
```

If you do not use option <code>-prof-src-root</code> during the <code>-prof-gen</code> phase, by default, the <code>-prof-use</code> compilation can only find the profile data if the file is compiled in the <code>c:\user1\feature</code> foo\my project\common directory.

#### See Also

```
prof-gen, Qprof-gen compiler option
prof-use, Qprof-use compiler option
prof-src-dir, Qprof-src-dir compiler option
prof-src-root-cwd, Qprof-src-root-cwd compiler option
```

## prof-src-root-cwd, Qprof-src-root-cwd

Lets you use relative directory paths when looking up profile data and specifies the current working directory as the base.

### **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: -prof-src-root-cwd

Windows: /Qprof-src-root-cwd

Arguments

None

Default

OFF The setting of relevant options determines the path used when looking up profile data records.

### Description

This option lets you use relative directory paths when looking up profile data in .dpi files. It specifies the current working directory as the base. To use this option, you must also specify option -prof-use (Linux and Mac OS) or /Oprof-use (Windows).

This option is available during the following phases of compilation:

- Linux and Mac OS X: -prof-gen and -prof-use phases
- Windows: /Qprof-gen and /Qprof-use phases

When this option is specified during the <code>-prof-gen</code> or <code>/Qprof-gen</code> phase, it stores information into the .dyn or .dpi file. Then, when .dyn files are merged together or the .dpi file is loaded, only the directory information below the root directory is used for forming the lookup key.

When this option is specified during the <code>-prof-use</code> or <code>/Qprof-use</code> phase, it specifies a root directory that replaces the root directory specified at the <code>-prof-gen</code> or <code>/Qprof-gen</code> phase for forming the lookup keys.

To be effective, this option or option <code>-prof-src-root</code> (Linux and Mac OS X) or <code>/Qprof-src-root</code> (Windows) must be specified during the <code>-prof-gen</code> or <code>/Qprof-gen</code> phase. In addition, if one of these options is not specified, absolute paths are used in the .dpi file.

### **Alternate Options**

None

### See Also

```
prof-gen, Qprof-gen compiler option
prof-use, Qprof-use compiler option
prof-src-dir, Qprof-src-dir compiler option
prof-src-root, Qprof-src-root compiler option
```

# prof-use, Qprof-use

Enables the use of profiling information during optimization.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: -prof-use[=arg]

-no-prof-use

Windows: /Qprof-use[:arg]

/Qprof-use-

### Arguments

arg Specifies additional instructions. Possible values

are:

weighted Tells the profmerge utility to

apply a weighting to the .dyn

file values when creating

the .dpi file to normalize the

data counts when the training

runs have differentexecution

durations. This argument only

has an effect when the compiler invokes the profmerge utility to create the .dpi file. This argument does not have an effect if the .dpi file was previously created without weighting.

[no]merge Enables or disables automatic invocation of the profmerge utility. The default is merge. Note that you cannot specify both weighted and nomerge. If you try to specify both values, a warning will be displayed and nomerge takes precedence.

default

Enables the use of profiling information during optimization. The profmerge utility is invoked by default. This value is the same as specifying -profuse (Linux and Mac OS X) or /Qprof-use (Windows) with no argument.

### Default

-no-Profiling information is not used during optimization.

prof-

use or

```
/Qprof-
use-
```

## Description

This option enables the use of profiling information (including function splitting and function grouping) during optimization. It enables option -fnsplit (Linux) or /Qfnsplit (Windows).

This option instructs the compiler to produce a profile-optimized executable and it merges available profiling output files into a popti.dpi file.

Note that there is no way to turn off function grouping if you enable it using this option.

To set the hotness threshold for function grouping and function ordering, use option -prof-hotness-threshold (Linux) or /Qprof-hotness-threshold (Windows).

### **Alternate Options**

None

### See Also

prof-hotness-threshold, Qprof-hotness-threshold compiler option
Optimizing Applications:

Basic PGO Options

Example of Profile-Guided Optimization

## rcd, Qrcd

Enables fast float-to-integer conversions.

### **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64 architectures

## Syntax

Linux and Mac OS X: -rcd

Windows: /Qrcd

### Arguments

None

#### Default

OFF Floating-point values are truncated when a conversion to an integer is involved. On Windows, this is the same as specifying /QIfist-.

## Description

This option enables fast float-to-integer conversions. It can improve the performance of code that requires floating-point-to-integer conversions. The system default floating-point rounding mode is round-to-nearest. However, the Fortran language requires floating-point values to be truncated when a conversion to an integer is involved. To do this, the compiler must change the rounding mode to truncation before each floating-point-to-integer conversion and change it back afterwards.

This option disables the change to truncation of the rounding mode for all floating-point calculations, including floating point-to-integer conversions. This option can improve performance, but floating-point conversions to integer will not conform to Fortran semantics.

### **Alternate Options**

Linux and Mac OS X: None

Windows: /QIfist

### rct, Qrct

Sets the internal FPU rounding control to Truncate.

IDE Equivalent
----------------

None

#### **Architectures**

IA-32, Intel® 64 architectures

# Syntax

Linux and Mac OS X: -rct

Windows: /Qrct

## Arguments

None

#### Default

OFF The compiler uses the default setting for the FPU rounding control.

## Description

This option sets the internal FPU rounding control to Truncate.

# **Alternate Options**

Linux and Mac OS X: None

Windows: /rounding-mode:chopped

# safe-cray-ptr, Qsafe-cray-ptr

Tells the compiler that Cray\* pointers do not alias other variables.

# **IDE Equivalent**

Windows: Data > Assume Cray Pointers Do Not Share Memory Locations

Linux: None

Mac OS X: Data > Assume Cray Pointers Do Not Share Memory Locations

## Architectures

IA-32, Intel® 64, IA-64 architectures

# Syntax

```
Linux and Mac OS X: -safe-cray-ptr
```

Windows: /Qsafe-cray-ptr

## Arguments

None

#### Default

OFF The compiler assumes that Cray pointers alias other variables.

## Description

This option tells the compiler that Cray pointers do not alias (that is, do not specify sharing memory with) other variables.

## **Alternate Options**

None

## Example

# Consider the following:

```
pointer (pb, b)
pb = getstorage()
do i = 1, n
b(i) = a(i) + 1
enddo
```

By default, the compiler assumes that b and a are aliased. To prevent such an assumption, specify the <code>-safe-cray-ptr</code> (Linux and Mac OS X) or <code>/Qsafe-cray-ptr</code> (Windows) option, and the compiler will treat b(i) and a(i) as independent of each other.

However, if the variables are intended to be aliased with Cray pointers, using the option produces incorrect results. In the following example, you should not use the option:

```
pointer (pb, b)
pb = loc(a(2))
do i=1, n
b(i) = a(i) +1
enddo
```

## save, Qsave

Causes variables to be placed in static memory.

## **IDE Equivalent**

Windows: **Data > Local Variable Storage** 

Linux: None

Mac OS X: Data > Local Variable Storage

### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -save

Windows: /Qsave

## Arguments

None

#### Default

```
-auto- Scalar variables of intrinsic types INTEGER, REAL,
scalar COMPLEX, and LOGICAL are allocated to the run-time stack.
or Note that if option recursive, -openmp (Linux and Mac OS
/Qauto- X), or /Qopenmp (Windows) is specified, the default is -
scalar automatic (Linux) or /Qauto (Windows).
```

## Description

This option saves all variables in static allocation except local variables within a recursive routine and variables declared as AUTOMATIC.

If you want all local, non-SAVEd variables to be allocated to the run-time stack, specify option automatic.

# **Alternate Options**

```
Linux and Mac OS X: -noautomatic, -noauto
```

```
Windows: /noautomatic, /noauto, /4Na
```

#### See Also

```
automatic compiler option
auto_scalar compiler option
```

# save-temps, Qsave-temps

Tells the compiler to save intermediate files created during compilation.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

```
Linux and Mac OS X: -save-temps
```

```
-no-save-temps
```

Windows: /Qsave-temps

/Qsave-temps-

## Arguments

None

### Default

Linux and Mac OS X: -no-save-

temps

Windows: .obj files are saved

On Linux and Mac OS X systems, the compiler deletes intermediate files after compilation is completed. On Windows systems, the compiler saves only intermediate object files after compilation is completed.

## Description

This option tells the compiler to save intermediate files created during compilation. The names of the files saved are based on the name of the source file; the files are saved in the current working directory.

If -save-temps or /Qsave-temps is specified, the following occurs:

- The object .o file (Linux and Mac OS X) or .obj file (Windows) is saved.
- The assembler .s file (Linux and Mac OS X) or .asm file (Windows) is saved if you specified -use-asm (Linux or Mac OS X) or /Quse-asm (Windows).
- The .i or .i90 file is saved if the fpp preprocessor is invoked.

If -no-save-temps is specified on Linux or Mac OS X systems, the following occurs:

- The .o file is put into /tmp and deleted after calling ld.
- The preprocessed file is not saved after it has been used by the compiler.

If /Qsave-temps- is specified on Windows systems, the following occurs:

- The .obj file is not saved after the linker step.
- The preprocessed file is not saved after it has been used by the compiler.



This option only saves intermediate files that are normally created during compilation.

#### **Alternate Options**

None

## Example

If you compile program  $my\_foo.F$  on a Linux or Mac OS X system and you specify option -save-temps and option -use-asm, the compilation will produce  $files my\_foo.o, my\_foo.s$ , and  $my\_foo.i$ .

If you compile program  $my\_foo.fpp$  on a Windows system and you specif option /Qsave-temps and option /Quse-asm, the compilation will produce files  $my\_foo.obj$ ,  $my\_foo.asm$ , and  $my\_foo.i$ .

# scalar-rep, Qscalar-rep

Enables scalar replacement performed during loop transformation.

# **IDE Equivalent**

None

### Architectures

IA-32 architecture

## **Syntax**

Linux and Mac OS X: -scalar-rep

-no-scalar-rep

Windows: /Qscalar-rep

/Qscalar-rep-

## Arguments

None

#### Default

```
-no- Scalar replacement is not performed during loop scalar-rep transformation.

or/Qscalar-
rep-
```

## Description

This option enables scalar replacement performed during loop transformation. To use this option, you must also specify O3.

# **Alternate Options**

None

## See Also

O compiler option

# Qsfalign

Specifies stack alignment for functions.

# **IDE Equivalent**

None

#### Architectures

IA-32 architecture

## **Syntax**

Linux and Mac OS X: None

Windows: /Qsfalign[n]

## Arguments

*n* Is the byte size of aligned variables. Possible

values are:

8 Specifies that alignment should

occur for functions with 8-byte

aligned variables. At this

setting the compiler aligns the stack to 16 bytes if there is any

16-byte or 8-byte data on the stack. For 8-byte data, the compiler only aligns the stack if the alignment will produce a performance advantage.

16

Specifies that alignment should occur for functions with 16-byte aligned variables. At this setting, the compiler only aligns the stack for 16-byte data. No attempt is made to align for 8-byte data.

#### Default

/Qsfalign8 Alignment occurs for functions with 8-byte aligned variables.

## Description

This option specifies stack alignment for functions. It lets you disable the normal optimization that aligns a stack for 8-byte data.

If you do not specify n, stack alignment occurs for all functions. If you specify /Qsfalign-, no stack alignment occurs for any function.

## **Alternate Options**

None

## sox, Qsox

Tells the compiler to save the compiler options and version number in the executable.

# **IDE Equivalent**

None

### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -sox

-no-sox

Windows: /Qsox

/Qsox-

### Arguments

None

#### Default

-no-sox The compiler does not save the compiler options and

or/Qsox- version number in the executable.

### Description

This option tells the compiler to save the compiler options and version number in the executable. The size of the executable on disk is increased slightly by the inclusion of these information strings.

This option forces the compiler to embed in each object file or assembly output a string that contains information about the compiler version and compilation options for each source file that has been compiled. When you link the object files into an executable file, the linker places each of the information strings into the header of the executable. It is then possible to use a tool, such as a strings utility, to determine what options were used to build the executable file.

If -no-sox or /Qsox- is specified, this extra information is not put into the object or assembly output generated by the compiler.

## **Alternate Options**

None

## tcheck, Qtcheck

Enables analysis of threaded applications.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux: -tcheck

Mac OS X: None

Windows: /Qtcheck

### Arguments

None

#### Default

OFF Threaded applications are not instrumented by the compiler for analysis by Intel® Thread Checker.

## Description

This option enables analysis of threaded applications.

To use this option, you must have Intel® Thread Checker installed, which is one of the Intel® Threading Analysis Tools. If you do not have this tool installed, the compilation will fail. Remove the -tcheck (Linux) or /Qtcheck (Windows) option from the command line and recompile.

For more information about Intel® Thread Checker (including an evaluation copy), open the page associated with threading tools at <a href="Intel® Software Development">Intel® Software Development</a>
<a href="Products">Products</a>.

## **Alternate Options**

None

## tcollect, Qtcollect

Inserts instrumentation probes calling the Intel® Trace Collector API.

## **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux: -tcollect[lib]

Mac OS X: None

Windows: /Qtcollect[:lib]

## Arguments

lib Is one of the Intel® Trace Collector libraries; for

example, VT, VTcs, VTmc, or VTfs. If you do not

specify *lib*, the default library is VT.

### Default

OFF Instrumentation probes are not inserted into compiled applications.

### Description

This option inserts instrumentation probes calling the Intel® Trace Collector API. To use this option, you must have the Intel® Trace Collector installed and set up through one of its set-up scripts. This tool is a component of the Intel® Trace Analyzer and Collector.

This option provides a flexible and convenient way of instrumenting functions of a compiled application. For every function, the entry and exit points are instrumented at compile time to let the Intel® Trace Collector record functions beyond the default MPI calls. For non-MPI applications (for example, threaded or serial), you must ensure that the Intel® Trace Collector is properly initialized (VT\_initialize/VT\_init).



Be careful with full instrumentation because this feature can produce very large trace files.

For more details, see the Intel® Trace Collector User Guide.

## **Alternate Options**

None

#### See Also

tcollect-filter, Qtcollect-filter compiler option

# tcollect-filter, Qtcollect-filter

Lets you enable or disable the instrumentation of specified functions.

### **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux: -tcollect-filter file

Mac OS X: None

Windows: /Qtcollect-filter:file

### Arguments

file

Is a configuration file that lists filters, one per line. Each filter consists of a regular expression string and a switch. Strings with leading or trailing white spaces must be quoted. Other strings do not have to be quoted. The switch value can be ON, on, OFF, or off.

## Default

OFF Functions are not instrumented. However, if option -tcollect (Linux) or /Qtcollect (Windows) is specified, the filter setting is ".\* ON" and all functions get instrumented.

## Description

This option lets you enable or disable the instrumentation of specified functions. During instrumentation, the regular expressions in the file are matched against the function names. The switch specifies whether matching functions are to be instrumented or not. Multiple filters are evaluated from top to bottom with increasing precedence.

The names of the functions to match against are formatted as follows:

• The source file name is followed by a colon-separated function name. Source file names should contain the full path, if available. For example:

```
/home/joe/src/file.f:FOO_bar
```

• Classes and function names are separated by double colons. For example:

```
/home/joe/src/file.fpp:app::foo::bar
```

You can use option <code>-opt-report</code> (Linux) or <code>/Qopt-report</code> (Windows) to get a full list of file and function names that the compiler recognizes from the compilation unit. This list can be used as the basis for filtering in the configuration file.

To use this option, you must have the Intel® Trace Collector installed and set up through one of its set-up scripts. This tool is a component of the Intel® Trace Analyzer and Collector.

For more details, see the Intel® Trace Collector User Guide.

## **Alternate Options**

None

Consider the following filters in a configuration file:

```
'.*' OFF '.*vector.*' ON
```

The above will cause instrumentation of only those functions having the string 'vector' in their names. No other function will be instrumented. Note that reversing the order of the two lines will prevent instrumentation of all functions.

To get a list of the file or routine strings that can be matched by the regular expression filters, generate an optimization report with tcollect information. For example:

```
Windows OS: ifort /Qtcollect /Qopt-report /Qopt-report-phase tcollect
Linux OS: ifort -tcollect -opt-report -opt-report-phase tcollect
```

### See Also

tcollect, Qtcollect compiler option

## tprofile, Qtprofile

Generates instrumentation to analyze multi-threading performance.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux: -tprofile

Mac OS X: None

Windows:

/Qtprofile

Arguments

None

Default

OFF Instrumentation is not generated by the compiler for analysis by

Intel® Thread Profiler.

Description

This option generates instrumentation to analyze multi-threading performance.

To use this option, you must have Intel® Thread Profiler installed, which is one of the Intel® Threading Analysis Tools. If you do not have this tool installed, the compilation will fail. Remove the -tprofile (Linux) or /Qtprofile (Windows)

option from the command line and recompile.

For more information about Intel® Thread Profiler (including an evaluation copy), open the page associated with threading tools at Intel® Software Development

Products.

**Alternate Options** 

None

ftrapuv, Qtrapuv

Initializes stack local variables to an unusual value to aid error detection.

IDE Equivalent

Windows: Data > Initialize stack variables to an unusual value

Linux: None

Mac OS X: Run-Time > Initialize Stack Variables to an Unusual Value

Architectures

601

IA-32, Intel® 64, IA-64 architectures

**Syntax** 

Linux and Mac OS X: -ftrapuv

Windows: /Qtrapuv

Arguments

None

Default

OFF The compiler does not initialize local variables.

Description

This option initializes stack local variables to an unusual value to aid error detection. Normally, these local variables should be initialized in the application. The option sets any uninitialized local variables that are allocated on the stack to a value that is typically interpreted as a very large integer or an invalid address. References to these variables are then likely to cause run-time errors that can help you detect coding errors.

This option sets option -g (Linux and Mac OS X) and /Zi or /Z7 (Windows).

**Alternate Options** 

None

See Also

g, Zi, Z7 compiler options

unroll, Qunroll

Tells the compiler the maximum number of times to unroll loops.

**IDE Equivalent** 

Windows: Optimization > Loop Unroll Count

602

Linux: None

Mac OS X: Optimization > Loop Unroll Count

### Architectures

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: -unroll[=n]

Windows: /Qunroll[:n]

### Arguments

*n* Is the maximum number of times a loop can be

unrolled. To disable loop enrolling, specify 0.

On systems using IA-64 architecture, you can only

specify a value of 0.

#### Default

-unroll The compiler uses default heuristics when unrolling loops.
or/Qunroll

## Description

This option tells the compiler the maximum number of times to unroll loops. If you do not specify n, the optimizer determines how many times loops can be unrolled.

# **Alternate Options**

Linux and Mac OS X: -funroll-loops

Windows: /unroll

## See Also

Optimizing Applications: Loop Unrolling

# unroll-aggressive, Qunroll-aggressive

Determines whether the compiler uses more aggressive unrolling for certain loops.

### **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -unroll-aggressive

-no-unroll-aggressive

Windows: /Qunroll-aggressive

/Qunroll-aggressive-

### Arguments

None

### Default

```
-no-unroll-aggressive The compiler uses default heuristics when or/Qunroll- unrolling loops.

aggressive-
```

### Description

This option determines whether the compiler uses more aggressive unrolling for certain loops. The positive form of the option may improve performance.

On IA-32 architecture and Intel® 64 architecture, this option enables aggressive, complete unrolling for loops with small constant trip counts.

On IA-64 architecture, this option enables additional complete unrolling for loops that have multiple exits or outer loops that have a small constant trip count.

Alternate Options
None
uppercase, Quppercase
See <u>names</u> .
use-asm, Quse-asm
Tells the compiler to produce objects through the assembler.
IDE Equivalent
None
Architectures
-use-asm: IA-32 architecture, Intel® 64 architecture, IA-64 architecture
/Quse-asm: IA-64 architecture
Syntax
Linux and Mac OS X: -use-asm
-no-use-asm
Windows: /Quse-asm
/Quse-asm-
Arguments
None
Default
-no-use- The compiler produces objects directly.
asm
or/Quse-

# Description

asm-

This option tells the compiler to produce objects through the assembler.

# **Alternate Options**

None

## **Quse-msasm-symbols**

Tells the compiler to use a dollar sign ("\$") when producing symbol names.

# IDE Equivalent

None

#### Architectures

IA-32, Intel® 64 architectures

# Syntax

Linux and Mac OS X: None

Windows: /Quse-msasm-symbols

## Arguments

None

## Default

OFF The compiler uses a period (".") when producing symbol names

## Description

This option tells the compiler to use a dollar sign ("\$") when producing symbol names.

Use this option if you require symbols in your .asm files to contain characters that are accepted by the MS assembler.

# **Alternate Options**

None

606

# **Quse-vcdebug**

Tells the compiler to issue debug information compatible with the Visual C++ debugger.

## **IDE Equivalent**

None

### **Architectures**

IA-32 architecture

# **Syntax**

Linux and Mac OS X: None

Windows: /Quse-vcdebug

# Arguments

None

### Default

OFF Debug information is issued that is compatible with Fortran debuggers.

# Description

This option tells the compiler to issue debug information compatible with the Visual C++ debugger. It prevents the compiler from issuing the extended information used by Fortran debuggers.

# **Alternate Options**

None

## Qvc

Specifies compatibility with Microsoft\* Visual C++ or Microsoft\* Visual Studio.

# **IDE Equivalent**

None

## Architectures

IA-32, Intel® 64 architectures

# Syntax

Linux and Mac OS X: None

Windows: /Qvc7.1

/Qvc8 /Qvc9

# Arguments

None

## Default

varies When the compiler is installed, it detects which version of Visual Studio is on your system. Qvc defaults to the form of the option that is compatible with that version. When multiple versions of Visual Studio are installed, the compiler installation lets you select which version you want to use. In this case, Qvc defaults to the version you choose.

# Description

This option specifies compatibility with Visual C++ or Visual Studio.

Option	Description
/Qvc7.1	Specifies compatibility with Microsoft*
	Visual Studio .NET 2003.
/Qvc8	Specifies compatibility with Microsoft*
	Visual Studio 2005.

Option	Description
/Qvc9	Specifies compatibility with Microsoft* Visual Studio 2008.
Alternate Options	
None	
vec, Qvec	
Enables or disables	s vectorization and transformations enabled for vectorization.
IDE Equivalent None	
Architectures	
IA-32, Intel® 64 ard	chitectures
Syntax	
Linux and Mac OS	X: -vec
	-no-vec
Windows:	/Qvec
	/Qvec-
Arguments	
None	
Default	
-vec	Vectorization is enabled.
or/Qvec	

Description

This option enables or disables vectorization and transformations enabled for vectorization.

To disable vectorization and transformations enabled for vectorization, specify – no-vec (Linux and Mac OS X) or /Qvec- (Windows).

# **Alternate Options**

None

# vec-guard-write, Qvec-guard-write

Tells the compiler to perform a conditional check in a vectorized loop.

# **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64 architectures

# Syntax

Linux and Mac OS X: -vec-guard-write

-no-vec-guard-write

Windows: /Qvec-guard-write

/Qvec-guard-write-

## **Arguments**

None

# Default

-no-vec-guard-write The compiler uses default heuristics when or/Qvec-guard-write- checking vectorized loops.

# Description

This option tells the compiler to perform a conditional check in a vectorized loop.

This checking avoids unnecessary stores and may improve performance.

# **Alternate Options**

None

# vec-report, Qvec-report

Controls the diagnostic information reported by the vectorizer.

## **IDE Equivalent**

Windows: Compilation Diagnostics > Vectorizer Diagnostic Level

Linux: None

Mac OS X: Diagnostics > Vectorizer Diagnostic Report

### Architectures

IA-32, Intel® 64 architectures

# Syntax

Linux and Mac OS X: -vec-report[n]

Windows: /Qvec-report[n]

## Arguments

*n* Is a value denoting which diagnostic messages to

report. Possible values are:

O Tells the vectorizer to report no

diagnostic information.

1 Tells the vectorizer to report on

vectorized loops.

2 Tells the vectorizer to report on

vectorized and non-vectorized

	loops.
3	Tells the vectorizer to report on
	vectorized and non-vectorized
	loops and any proven or
	assumed data dependences.
4	Tells the vectorizer to report on
	non-vectorized loops.
5	Tells the vectorizer to report on
	non-vectorized loops and the
	reason why they were not
	vectorized.

#### Default

-vec-	If the vectorizer has been enabled, it reports diagnostics
report1	on vectorized loops.
or/Qvec-	
report1	

# Description

This option controls the diagnostic information reported by the vectorizer. The vectorizer report is sent to stdout.

If you do not specify n, it is the same as specifying -vec-report1 (Linux and Mac OS X) or /Qvec-report1 (Windows).

The vectorizer is enabled when certain compiler options are specified, such as option -ax or -x (Linux and Mac OS X), option /Qax or /Qx (Windows), option -ax or -ax or -ax has SSE or -ax has SSE2 (Linux and Mac OS X), option /ax hitecture: SSE or /ax hitecture: SSE2 (Windows).

If this option is specified from within the IDE, the report is included in

If this option is specified from within the IDE, the report is included in the build log if the Generate Build Logs option is selected.

## **Alternate Options**

None

# x, Qx

Tells the compiler to generate optimized code specialized for the Intel processor that executes your program.

# IDE Equivalent

Windows: Code Generation > Intel Processor-Specific Optimization

Optimization > Use Intel(R) Processor Extensions

Linux: None

Mac OS X: Code Generation > Intel Processor-Specific Optimization

### Architectures

IA-32, Intel® 64 architectures

# **Syntax**

Linux and Mac OS X: -xprocessor

Windows: /Oxprocessor

# Arguments

processor Indicates the processor for which code is

generated. Many of the following descriptions refer to Intel® Streaming SIMD Extensions (Intel® SSE) and Supplemental Streaming SIMD Extensions

(Intel® SSSE). Possible values are:

Host Can generate instructions for

the highest instruction set and

processor available on the

compilation host.

- SSE4.2 Can generate Intel® SSE4

  Efficient Accelerated String and

  Text Processing instructions

  supported by Intel® Core™ i7

  processors. Can generate

  Intel® SSE4 Vectorizing

  Compiler and Media

  Accelerator, Intel® SSSE3,

  SSE3, SSE2, and SSE

  instructions and it can optimize

  for the Intel® Core™ processor

  family.
- Vectorizing Compiler and
  Media Accelerator instructions
  for Intel processors. Can
  generate Intel® SSSE3, SSE3,
  SSE2, and SSE instructions
  and it can optimize for Intel®
  45nm Hi-k next generation
  Intel® Core™
  microarchitecture. This
  replaces value S, which is
  deprecated.
- SSE3\_ATOM Can generate MOVBE
  instructions for Intel processors
  and it can optimize for the
  Intel® Atom™ processor and
  Intel® Centrino® Atom™
  Processor Technology.

SSSE3 Can generate Intel® SSSE3,

SSE3, SSE2, and SSE

instructions for Intel processors

and it can optimize for the

Intel® Core™2 Duo processor

family. This replaces value T,

which is deprecated.

SSE3 Can generate Intel® SSE3,

SSE2, and SSE instructions for

Intel processors and it can

optimize for processors based

on Intel® Core™

microarchitecture and Intel

NetBurst® microarchitecture.

This replaces value P, which is

deprecated.

SSE2 Can generate Intel® SSE2 and

SSE instructions for Intel

processors, and it can optimize

for Intel® Pentium® 4

processors, Intel® Pentium® M

processors, and Intel® Xeon®

processors with Intel® SSE2.

This value is not available on

Mac OS X systems. This

replaces value N, which is

deprecated.

**Default** 

Windows systems: For more information on the default values, see

Arguments above, option m (Linux and Mac OS X) and

option arch (Windows).

Linux systems: -

/arch:SSE2

msse2

Mac OS X systems

using IA-32

architecture: SSE3

Mac OS X systems

using Intel® 64

architecture: SSSE3

# Description

This option tells the compiler to generate optimized code specialized for the Intel processor that executes your program. It also enables optimizations in addition to Intel processor-specific optimizations. The specialized code generated by this option may run only on a subset of Intel processors.

This option can enable optimizations depending on the argument specified. For example, it may enable Intel® Streaming SIMD Extensions 4 (Intel® SSE4), Intel® Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), Intel® Streaming SIMD Extensions 2 (Intel® SSE2), or Intel® Streaming SIMD Extensions (Intel® SSE) instructions. The binaries produced by these values will run on Intel processors that support all of the features for the targeted processor. For example, binaries produced with SSE3 will run on an Intel® Core™ 2 Duo processor, because that processor completely supports all of the capabilities of the Intel® Pentium® 4 processor, which the SSE3 value targets. Specifying the SSSE3 value has the potential of using more features and optimizations available to the Intel® Core™ 2 Duo processor.

Do not use *processor* values to create binaries that will execute on a processor that is not compatible with the targeted processor. The resulting program may fail with an illegal instruction exception or display other unexpected behavior. For

example, binaries produced with SSE3 may produce code that will not run on Intel® Pentium® III processors or earlier processors that do not support SSE2 instructions.

Compiling the function main() with any of the *processor* values produces binaries that display a fatal run-time error if they are executed on unsupported processors. For more information, see *Optimizing Applications*.

If you specify more than one *processor* value, code is generated for only the highest-performing processor specified. The highest-performing to lowest-performing *processor* values are: SSE4.2, SSE4.1, SSSE3, SSE3, SSE2. Note that *processor* value SSE3\_ATOM does not fit within this group.

Compiler options  ${\tt m}$  and  ${\tt arch}$  produce binaries that should run on processors not made by Intel that implement the same capabilities as the corresponding Intel processors.

Previous value O is deprecated and has been replaced by option -msse3 (Linux and Mac OS X) and option /arch: SSE3 (Windows).

Previous values W and K are deprecated. The details on replacements are as follows:

- Mac OS X systems: On these systems, there is no exact replacement for W or K. You can upgrade to the default option -msse3 (IA-32 architecture) or option -mssse3 (Intel® 64 architecture).
- Windows and Linux systems: The replacement for W is -msse2 (Linux) or /arch: SSE2 (Windows). There is no exact replacement for K. However, on Windows systems, /QxK is interpreted as /arch: IA32; on Linux systems, -xK is interpreted as -mia32. You can also do one of the following:
- Upgrade to option -msse2 (Linux) or option /arch: SSE2 (Windows). This
  will produce one code path that is specialized for Intel® SSE2. It will not run
  on earlier processors
- Specify the two option combination -mia32 -axSSE2 (Linux) or /arch:IA32 /QaxSSE2 (Windows). This combination will produce an executable that runs on any processor with IA-32 architecture but with an additional specialized Intel® SSE2 code path.

The -x and /Qx options enable additional optimizations not enabled with option -m or option /arch.

# **Alternate Options**

None

### See Also

ax, Qax compiler optioncompiler optionarch compiler option

# zero, Qzero

Initializes to zero all local scalar variables of intrinsic type INTEGER, REAL, COMPLEX, or LOGICAL that are saved but not yet initialized.

# IDE Equivalent

Windows: Data > Initialize Local Saved Scalars to Zero

Linux: None

Mac OS X: Data > Initialize Local Saved Scalars to Zero

### Architectures

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -zero

-nozero

Windows: /Qzero

/Qzero-

# Arguments

None

# Default

-nozero Local scalar variables are not initialized to zero.

or

/Qzero-

# Description

This option initializes to zero all local scalar variables of intrinsic type INTEGER, REAL, COMPLEX, or LOGICAL that are saved but not yet initialized.

Use -save (Linux and Mac OS X) or /Qsave (Windows) on the command line to make all local variables specifically marked as SAVE.

# **Alternate Options**

None

# See Also

save compiler option

r8, r16

See <u>real-size</u>.

# rcd, Qrcd

Enables fast float-to-integer conversions.

# **IDE Equivalent**

None

# Architectures

IA-32, Intel® 64 architectures

# **Syntax**

Linux and Mac OS X: -rcd

Windows:

/Qrcd

Arguments

None

Default

OFF Floating-point values are truncated when a conversion to an integer is involved. On Windows, this is the same as specifying

/QIfist-.

Description

This option enables fast float-to-integer conversions. It can improve the

performance of code that requires floating-point-to-integer conversions.

The system default floating-point rounding mode is round-to-nearest. However,

the Fortran language requires floating-point values to be truncated when a

conversion to an integer is involved. To do this, the compiler must change the

rounding mode to truncation before each floating-point-to-integer conversion and

change it back afterwards.

This option disables the change to truncation of the rounding mode for all

floating-point calculations, including floating point-to-integer conversions. This

option can improve performance, but floating-point conversions to integer will not

conform to Fortran semantics.

**Alternate Options** 

Linux and Mac OS X: None

Windows: /QIfist

rct, Qrct

Sets the internal FPU rounding control to Truncate.

**IDE Equivalent** 

620

Ν	or	ne
---	----	----

## **Architectures**

IA-32, Intel® 64 architectures

# **Syntax**

Linux and Mac OS X: -rct

Windows: /Qrct

# **Arguments**

None

## Default

OFF The compiler uses the default setting for the FPU rounding control.

# Description

This option sets the internal FPU rounding control to Truncate.

# **Alternate Options**

Linux and Mac OS X: None

Windows: /rounding-mode:chopped

## real-size

Specifies the default KIND for real and complex variables.

# **IDE Equivalent**

Windows: Data > Default Real KIND

Linux: None

Mac OS X: Data > Default Real KIND

## **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -real-size size
Windows: /real-size:size

# Arguments

size Is the size for real and complex variables. Possible values are: 32, 64, or 128.

# Default

real- Default real and complex variables are 4 bytes long size (REAL(KIND=4) and COMPLEX(KIND=4)).

# Description

This option specifies the default size (in bits) for real and complex variables.

Option	Description
real-size	Makes default real and complex variables 4 bytes long. REAL declarations are treated as single precision REAL (REAL(KIND=4)) and COMPLEX declarations are treated as COMPLEX (COMPLEX(KIND=4)).
real-size 64	Makes default real and complex variables 8 bytes long. REAL declarations are treated as DOUBLE PRECISION (REAL(KIND=8)) and COMPLEX declarations are treated as DOUBLE COMPLEX (COMPLEX(KIND=8)).
real-size 128	Makes default real and complex variables 16 bytes long. REAL declarations are treated as extended precision REAL (REAL(KIND=16)); COMPLEX and DOUBLE COMPLEX

Option	Description
	declarations are treated as extended precision COMPLEX
	(COMPLEX(KIND=16)).

These compiler options can affect the result type of intrinsic procedures, such as CMPLX, FLOAT, REAL, SNGL, and AIMAG, which normally produce single-precision REAL or COMPLEX results. To prevent this effect, you must explicitly declare the kind type for arguments of such intrinsic procedures.

For example, if real-size 64 is specified, the CMPLX intrinsic will produce a result of type DOUBLE COMPLEX (COMPLEX(KIND=8)). To prevent this, you must explicitly declare any real argument to be REAL(KIND=4), and any complex argument to be COMPLEX(KIND=4).

## **Alternate Options**

real-size 64 Linux and Mac OS X: -r8, -autodouble
Windows: /4R8, /Qautodouble

real-size 128 Linux and Mac OS X: -r16

Windows: /4R16

### recursive

Tells the compiler that all routines should be compiled for possible recursive execution.

# **IDE Equivalent**

Windows: Code Generation > Enable Recursive Routines

Linux: None

Mac OS X: Code Generation > Enable Recursive Routines

### Architectures

IA-32, Intel® 64, IA-64 architectures

Systems: Windows, Linux

Syntax

Linux and Mac OS X: -recursive

-norecursive

Windows: /recursive

/norecursive

# Arguments

None

## Default

norecursive Routines are not compiled for possible recursive execution.

# Description

This option tells the compiler that all routines should be compiled for possible recursive execution. It sets the automatic option.

# **Alternate Options**

None

## See Also

automatic compiler option

# reentrancy

Tells the compiler to generate reentrant code to support a multithreaded application.

# **IDE Equivalent**

Windows: Code Generation > Generate Reentrant Code

Linux: None

# Mac OS X: Code Generation > Generate Reentrant Code

### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -reentrancy keyword

-noreentrancy

Windows: /reentrancy:keyword

/noreentrancy

# Arguments

*keyword* Specifies details about the program. Possible values are:

none Tells the run-time library (RTL) that the

program does not rely on threaded or asynchronous reentrancy. The RTL will

not guard against such interrupts

inside its own critical regions. This is

the same as specifying noreentrancy.

async Tells the run-time library (RTL) that the

program may contain asynchronous

(AST) handlers that could call the RTL.

This causes the RTL to guard against

AST interrupts inside its own critical

regions.

threaded Tells the run-time library (RTL) that the

program is multithreaded, such as programs using the POSIX threads

library. This causes the RTL to use

thread locking to guard its own critical

regions.

### Default

noreentrancy The compiler does not generate reentrant code for applications.

# Description

This option tells the compiler to generate reentrant code to support a multithreaded application.

If you do not specify a keyword for reentrancy, it is the same as specifying reentrancy threaded.

Note that if option threads is specified, it sets option reentrancy threaded, since multithreaded code must be reentrant.

# **Alternate Options**

None

## See Also

threads compiler option

# RTCu

See check.

S

Causes the compiler to compile to an assembly file only and not link.

# **IDE Equivalent**

None

### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -S

Windows: /S

## Arguments

None

### Default

OFF Normal compilation and linking occur.

# Description

This option causes the compiler to compile to an assembly file only and not link. On Linux and Mac OS X systems, the assembly file name has a .s suffix. On Windows systems, the assembly file name has an .asm suffix.

# **Alternate Options**

Linux and Mac OS X: None

Windows: /Fa, /asmfile

### See Also

Fa compiler option

# safe-cray-ptr, Qsafe-cray-ptr

Tells the compiler that Cray\* pointers do not alias other variables.

# **IDE Equivalent**

Windows: Data > Assume Cray Pointers Do Not Share Memory Locations

Linux: None

Mac OS X: Data > Assume Cray Pointers Do Not Share Memory Locations

### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

```
Linux and Mac OS X: -safe-cray-ptr
Windows: /Qsafe-cray-ptr
```

# Arguments

None

#### Default

OFF The compiler assumes that Cray pointers alias other variables.

## Description

This option tells the compiler that Cray pointers do not alias (that is, do not specify sharing memory with) other variables.

# **Alternate Options**

None

### Example

Consider the following:

```
pointer (pb, b)
pb = getstorage()
do i = 1, n
b(i) = a(i) + 1
enddo
```

By default, the compiler assumes that b and a are aliased. To prevent such an assumption, specify the -safe-cray-ptr (Linux and Mac OS X) or /Qsafe-cray-ptr (Windows) option, and the compiler will treat b(i) and a(i) as independent of each other.

However, if the variables are intended to be aliased with Cray pointers, using the option produces incorrect results. In the following example, you should not use the option:

```
pointer (pb, b)
```

```
pb = loc(a(2))
do i=1, n
b(i) = a(i) +1
enddo
```

## save, Qsave

Causes variables to be placed in static memory.

## **IDE Equivalent**

Windows: **Data > Local Variable Storage** 

Linux: None

Mac OS X: **Data > Local Variable Storage** 

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -save

Windows: /Qsave

# Arguments

None

### Default

```
-auto- Scalar variables of intrinsic types INTEGER, REAL,
scalar COMPLEX, and LOGICAL are allocated to the run-time stack.
or Note that if option recursive, -openmp (Linux and Mac OS
/Qauto- X), or /Qopenmp (Windows) is specified, the default is -
scalar automatic (Linux) or /Qauto (Windows).
```

# Description

This option saves all variables in static allocation except local variables within a recursive routine and variables declared as AUTOMATIC.

If you want all local, non-SAVEd variables to be allocated to the run-time stack, specify option automatic.

# **Alternate Options**

Linux and Mac OS X: -noautomatic, -noauto

Windows: /noautomatic, /noauto, /4Na

#### See Also

```
automatic compiler option
auto_scalar compiler option
```

# save-temps, Qsave-temps

Tells the compiler to save intermediate files created during compilation.

# **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -save-temps

-no-save-temps

Windows: /Qsave-temps

/Qsave-temps-

# Arguments

None

# Default

Linux and Mac OS X: -no-save- On Linux and Mac OS X systems, the compiler deletes intermediate

Windows: .obj files are saved files after compilation is completed.

On Windows systems, the compiler saves only intermediate object files

after compilation is completed.

# Description

This option tells the compiler to save intermediate files created during compilation. The names of the files saved are based on the name of the source file; the files are saved in the current working directory.

If -save-temps or /Qsave-temps is specified, the following occurs:

- The object .o file (Linux and Mac OS X) or .obj file (Windows) is saved.
- The assembler .s file (Linux and Mac OS X) or .asm file (Windows) is saved if you specified -use-asm (Linux or Mac OS X) or /Quse-asm (Windows).
- The .i or .i90 file is saved if the fpp preprocessor is invoked.

If -no-save-temps is specified on Linux or Mac OS X systems, the following occurs:

- The .o file is put into /tmp and deleted after calling ld.
- The preprocessed file is not saved after it has been used by the compiler.

If /Qsave-temps- is specified on Windows systems, the following occurs:

- The .obj file is not saved after the linker step.
- The preprocessed file is not saved after it has been used by the compiler.



This option only saves intermediate files that are normally created during compilation.

## **Alternate Options**

None

# Example

If you compile program  $my\_foo.F$  on a Linux or Mac OS X system and you specify option -save-temps and option -use-asm, the compilation will produce  $files my\_foo.o, my\_foo.s$ , and  $my\_foo.i$ .

If you compile program  $my\_foo.fpp$  on a Windows system and you specif option /Qsave-temps and option /Quse-asm, the compilation will produce files  $my\_foo.obj$ ,  $my\_foo.asm$ , and  $my\_foo.i$ .

# scalar-rep, Qscalar-rep

Enables scalar replacement performed during loop transformation.

## **IDE Equivalent**

None

## Architectures

IA-32 architecture

# Syntax

Linux and Mac OS X: -scalar-rep

-no-scalar-rep

Windows: /Qscalar-rep

/Qscalar-rep-

## Arguments

None

## Default

```
-no- Scalar replacement is not performed during loop scalar-rep transformation.

or/Qscalar-
rep-
```

# Description

This option enables scalar replacement performed during loop transformation.	То
use this option, you must also specify 03.	

Alternate Options	
None	
See Also	
O compiler option	
shared	
Tells the compiler to	produce a dynamic shared object instead of an executable
IDE Equivalent	
None	
Architectures	
IA-32, Intel® 64, IA-64 architectures	
Syntax	
Linux:	-shared
Mac OS X:	None
Windows:	None
Arguments	
None	
Default	
OFF The compiler	produces an executable.

Description

This option tells the compiler to produce a dynamic shared object (DSO) instead of an executable. This includes linking in all libraries dynamically and passing – shared to the linker.

On systems using IA-32 architecture and Intel® 64 architecture, you must specify option fpic for the compilation of each object file you want to include in the shared library.

# **Alternate Options**

None

## See Also

dynamiclib compiler option
fpic compiler option
Xlinker compiler option

## shared-intel

Causes Intel-provided libraries to be linked in dynamically.

# **IDE Equivalent**

Windows: None

Linux: None

Mac OS X: Run-Time > Intel Runtime Libraries

# Architectures

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -shared-intel

Windows: None

## Arguments

None

# Default

OFF Intel libraries are linked in statically, with the exception of libguide on Linux\* and Mac OS\* X systems, where it is linked in dynamically.

# Description

This option causes Intel-provided libraries to be linked in dynamically. It is the opposite of -static-intel.



On Mac OS X systems, when you set "Intel Runtime Libraries" to "Dynamic", you must also set the DYLD\_LIBRARY\_PATH environment variable within Xcode or an error will be displayed.

# **Alternate Options**

Linux and Mac OS X: -i-dynamic (this is a deprecated option)

Windows: None

## See Also

static-intel compiler option

# shared-libgcc

Links the GNU libgcc library dynamically.

# **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux: -shared-libgcc Mac OS X: None Windows: None Arguments None Default -shared-The compiler links the libgcc library dynamically. libgcc Description This option links the GNU libger library dynamically. It is the opposite of option static-libgcc. This option is useful when you want to override the default behavior of the static option, which causes all libraries to be linked statically. **Alternate Options** None See Also static-libgcc compiler option source Tells the compiler to compile the file as a Fortran source file. **IDE Equivalent** None Architectures IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: None

Windows: /source:file

## Arguments

file Is the name of the file.

### Default

OFF Files that do not end in standard Fortran file extensions are not compiled as Fortran files.

## Description

This option tells the compiler to compile the file as a Fortran source file.

This option is useful when you have a Fortran file with a nonstandard file extension (that is, not one of .F, .FOR, or .F90).

This option assumes the file specified uses fixed source form. If the file uses free source form, you must also specify option free.

# **Alternate Options**

Linux and Mac OS X: -Tf file

Windows: /Tf file

# See Also

extfor compiler option
free compiler option

## sox, Qsox

Tells the compiler to save the compiler options and version number in the executable.

## **IDE Equivalent**

None

## **Architectures**

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -sox

-no-sox

Windows: /Qsox

/Qsox-

## Arguments

None

### Default

-no-sox The compiler does not save the compiler options and

or/Qsox- version number in the executable.

## Description

This option tells the compiler to save the compiler options and version number in the executable. The size of the executable on disk is increased slightly by the inclusion of these information strings.

This option forces the compiler to embed in each object file or assembly output a string that contains information about the compiler version and compilation options for each source file that has been compiled. When you link the object files into an executable file, the linker places each of the information strings into the header of the executable. It is then possible to use a tool, such as a strings utility, to determine what options were used to build the executable file.

If -no-sox or /Qsox- is specified, this extra information is not put into the object or assembly output generated by the compiler.

## **Alternate Options**

None

### stand

Tells the compiler to issue compile-time messages for nonstandard language elements.

# **IDE Equivalent**

Windows: Compilation Diagnostics > Warn For Nonstandard Fortran

Linux: None

Mac OS X: Compiler Diagnostics > Warn For Nonstandard Fortran

## Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -stand [keyword]

-nostand

Windows: /stand[:keyword]

/nostand

# Arguments

keyword Specifies the language to use as the standard. Possible values

are:

none Issue no messages for nonstandard

language elements.

f 90 Issue messages for language

elements that are not standard in

Fortran 90.

f 95 Issue messages for language

elements that are not standard in

Fortran 95.

lssue messages for language elements that are not standard in Fortran 2003.

## Default

nostand The compiler issues no messages for nonstandard language elements.

# Description

This option tells the compiler to issue compile-time messages for nonstandard language elements.

If you do not specify a keyword for stand, it is the same as specifying stand f95.

Option	Description
stand none	Tells the compiler to issue no messages for nonstandard
	language elements. This is the same as specifying nostand.
stand f90	Tells the compiler to issue messages for language elements that
	are not standard in Fortran 90.
stand f95	Tells the compiler to issue messages for language elements that
	are not standard in Fortran 95.
stand f03	Tells the compiler to issue messages for language elements that
	are not standard in Fortran 2003. This option is set if you specify
	warn stderrors.

# **Alternate Options**

stand none Linux and Mac OS X: -nostand
Windows: /nostand, /4Ns

stand f90 Linux and Mac OS X: -std90

Windows: /4Ys

stand f95 Linux and Mac OS X: -std95

Windows: None

stand f03 Linux and Mac OS X: -std03, -stand, -std

Windows: /stand

## See Also

warn compiler option

## static

Prevents linking with shared libraries.

# **IDE Equivalent**

None

### **Architectures**

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux: -static

Mac OS X: None

Windows: /static

# Arguments

None

# Default

static The compiler does not link with shared libraries.

# Description

This option prevents linking with shared libraries. It causes the executable to link all libraries statically.

# **Alternate Options**

None

## staticlib

Invokes the libtool command to generate static libraries.

# **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64 architectures

# **Syntax**

Linux: None

Mac OS X: -staticlib

Windows: None

# Arguments

None

### Default

OFF The compiler produces an executable.

# Description

This option invokes the libtool command to generate static libraries.

When passed this option, the compiler uses the libtool command to produce a static library instead of an executable when linking.

To build dynamic librar	ries, you should specify option	-dynamiclib or	libtool
-dynamic <objects< td=""><td>5&gt;.</td><td></td><td></td></objects<>	5>.		

## **Alternate Options**

None

#### See Also

dynamiclib compiler option

## static-intel

Causes Intel-provided libraries to be linked in statically.

## **IDE Equivalent**

Windows: None

Linux: None

Mac OS X: Run-Time > Intel Runtime Libraries

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -static-intel

Windows: None

## Arguments

None

#### Default

OFF Intel libraries are linked in statically, with the exception of libguide, which is linked in dynamically.

## Description

This option causes Intel-provided libraries to be linked in statically. It is the opposite of -shared-intel.

## **Alternate Options**

Linux and Mac OS X: i-static (this is a <u>deprecated</u> option)

Windows: None

#### See Also

shared-intel compiler option

## static-libgcc

Links the GNU libgcc library statically.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux: -static-libgcc

Mac OS X: None

Windows: None

## Arguments

None

## Default

OFF DEFAULT\_DESC

## Description

This option links the GNU libgcc library statically. It is the opposite of option libgcc.

This option is useful when you want to override the default behavior of the libgec option, which causes all libraries to be linked statically.

## **Alternate Options**

None

## See Also

shared-libgcc compiler option

std, std90, std95, std03

See stand.

## syntax-only

Tells the compiler to check only for correct syntax.

## **IDE Equivalent**

None

### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -syntax-only

Windows: /syntax-only

### Arguments

None

#### Default

OFF Normal compilation is performed.

## Description

This option tells the compiler to check only for correct syntax. It lets you do a quick syntax check of your source file.

Compilation stops after the source file has been parsed. No code is generated, no object file is produced, and some error checking done by the optimizer is bypassed.

Warnings and messages appear on stderr.

## **Alternate Options**

Linux: -y, -fsyntax-only, -syntax (this is a deprecated option)

Mac OS X: -y, -fsyntax-only

Windows: /Zs

## Т

Tells the linker to read link commands from a file.

## IDE Equivalent

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

Linux:	-Tfile
Mac OS X:	None
Windows:	None
Arguments	
file	Is the name of the file.
Default  OFF The linker doe	es not read link commands from a file.
OFF THE IIIKEI GOE	es not read link commands nom a me.
Description	
This option tells the li	nker to read link commands from a file.
Alternate Options	
None	
tcheck, Qtcheck	
Enables analysis of threaded applications.	
IDE Equivalent	
None	
Architectures	
IA-32, Intel® 64, IA-6	4 architectures
Syntax	

Syntax

Linux:

Mac OS X:

Windows:

-tcheck

/Qtcheck

None

**Arguments** 

None

Default

OFF Threaded applications are not instrumented by the compiler for analysis by Intel® Thread Checker.

Description

This option enables analysis of threaded applications.

To use this option, you must have Intel® Thread Checker installed, which is one of the Intel® Threading Analysis Tools. If you do not have this tool installed, the compilation will fail. Remove the -tcheck (Linux) or /Qtcheck (Windows) option from the command line and recompile.

For more information about Intel® Thread Checker (including an evaluation copy), open the page associated with threading tools at <a href="Intel® Software Development">Intel® Software Development</a>
Products.

**Alternate Options** 

None

tcollect, Qtcollect

Inserts instrumentation probes calling the Intel® Trace Collector API.

**IDE Equivalent** 

None

Architectures

IA-32, Intel® 64, IA-64 architectures

Syntax

Linux: -tcollect[lib]

648

Mac OS X: None

Windows: /Qtcollect[:lib]

#### Arguments

lib Is one of the Intel® Trace Collector libraries; for

example, VT, VTcs, VTmc, or VTfs. If you do not

specify *lib*, the default library is VT.

#### Default

OFF Instrumentation probes are not inserted into compiled applications.

### Description

This option inserts instrumentation probes calling the Intel® Trace Collector API. To use this option, you must have the Intel® Trace Collector installed and set up through one of its set-up scripts. This tool is a component of the Intel® Trace Analyzer and Collector.

This option provides a flexible and convenient way of instrumenting functions of a compiled application. For every function, the entry and exit points are instrumented at compile time to let the Intel® Trace Collector record functions beyond the default MPI calls. For non-MPI applications (for example, threaded or serial), you must ensure that the Intel® Trace Collector is properly initialized (VT\_initialize/VT\_init).

# A Caution

Be careful with full instrumentation because this feature can produce very large trace files.

For more details, see the Intel® Trace Collector User Guide.

## **Alternate Options**

None

## See Also

tcollect-filter, Qtcollect-filter compiler option

## tcollect-filter, Qtcollect-filter

Lets you enable or disable the instrumentation of specified functions.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux: -tcollect-filter file

Mac OS X: None

Windows: /Qtcollect-filter:file

### Arguments

file Is a configuration file that lists filters, one per line.

Each filter consists of a regular expression string and a switch. Strings with leading or trailing white spaces must be quoted. Other strings do not have to be quoted. The switch value can be ON, on,

OFF, or off.

#### Default

OFF Functions are not instrumented. However, if option -tcollect (Linux) or /Qtcollect (Windows) is specified, the filter setting is ".\* ON" and all functions get instrumented.

## Description

This option lets you enable or disable the instrumentation of specified functions. During instrumentation, the regular expressions in the file are matched against the function names. The switch specifies whether matching functions are to be instrumented or not. Multiple filters are evaluated from top to bottom with increasing precedence.

The names of the functions to match against are formatted as follows:

 The source file name is followed by a colon-separated function name. Source file names should contain the full path, if available. For example:

```
/home/joe/src/file.f:FOO_bar
```

• Classes and function names are separated by double colons. For example:

```
/home/joe/src/file.fpp:app::foo::bar
```

You can use option <code>-opt-report</code> (Linux) or <code>/Qopt-report</code> (Windows) to get a full list of file and function names that the compiler recognizes from the compilation unit. This list can be used as the basis for filtering in the configuration file.

To use this option, you must have the Intel® Trace Collector installed and set up through one of its set-up scripts. This tool is a component of the Intel® Trace Analyzer and Collector.

For more details, see the Intel® Trace Collector User Guide.

#### **Alternate Options**

#### None

Consider the following filters in a configuration file:

```
'.*' OFF '.*vector.*' ON
```

The above will cause instrumentation of only those functions having the string 'vector' in their names. No other function will be instrumented. Note that reversing the order of the two lines will prevent instrumentation of all functions.

To get a list of the file or routine strings that can be matched by the regular expression filters, generate an optimization report with tcollect information. For example:

```
Windows OS: ifort /Qtcollect /Qopt-report /Qopt-report-phase tcollect
Linux OS: ifort -tcollect -opt-report -opt-report-phase tcollect
```

#### See Also

tcollect, Qtcollect compiler option

Tf

See source.

#### threads

Tells the linker to search for unresolved references in a multithreaded run-time library.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -threads

-nothreads

Windows: /threads

/nothreads

## Arguments

None

#### Default

Systems On systems using IA-32 architecture and IA-64 architecture, using Intel® the linker does not search for unresolved references in a mutithreaded run-time library. On systems using Intel® 64 architecture: architectures, it does.

threads

Systems

using IA-32

architecture
and IA-64
architecture:
nothreads

## Description

This option tells the linker to search for unresolved references in a multithreaded run-time library.

This option sets option reentrancy threaded.

Windows systems: The following table shows which options to specify for a multithreaded run-time library.

Type of Library	Options Required	Alternate Option
Multithreaded	/libs:static	/MT
	/threads	
Debug multithreaded	/libs:static	/MTd
	/threads	
	/dbglibs	
Multithreaded DLLs	/libs:dll	/MD
	/threads	
Multithreaded debug DLLs	/libs:dll	/MDd
	/threads	
	/dbglibs	

## **Alternate Options**

None

#### See Also

Building Applications: Specifying Consistent Library Types; Programming with Mixed Languages Overview

## tprofile, Qtprofile

Generates instrumentation to analyze multi-threading performance.

## **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux: -tprofile

Mac OS X: None

Windows: /Qtprofile

## Arguments

None

#### Default

OFF Instrumentation is not generated by the compiler for analysis by Intel® Thread Profiler.

## Description

This option generates instrumentation to analyze multi-threading performance. To use this option, you must have Intel® Thread Profiler installed, which is one of the Intel® Threading Analysis Tools. If you do not have this tool installed, the compilation will fail. Remove the <code>-tprofile</code> (Linux) or <code>/Qtprofile</code> (Windows) option from the command line and recompile.

For more information about Intel® Thread Profiler (including an evaluation copy), open the page associated with threading tools at <a href="Intel® Software Development">Intel® Software Development</a>
<a href="Products">Products</a>.

## **Alternate Options**

None

654

#### traceback

Tells the compiler to generate extra information in the object file to provide source file traceback information when a severe error occurs at run time.

#### **IDE Equivalent**

Windows: Run-time > Generate Traceback Information

Linux: None

Mac OS X: Run-time > Generate Traceback Information

## Architectures

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: -traceback

-notraceback

Windows: /traceback

/notraceback

#### Arguments

None

#### Default

notraceback No extra information is generated in the object file to produce traceback information.

#### Description

This option tells the compiler to generate extra information in the object file to provide source file traceback information when a severe error occurs at run time. When the severe error occurs, source file, routine name, and line number correlation information is displayed along with call stack hexadecimal addresses (program counter trace).

Note that when a severe error occurs, advanced users can also locate the cause of the error using a map file and the hexadecimal addresses of the stack displayed when the error occurs.

This option increases the size of the executable program, but has no impact on run-time execution speeds.

It functions independently of the debug option.

On Windows systems, traceback sets the /Oy- option, which forces the compiler to use EBP as the stack frame pointer.

On Windows systems, the linker places the traceback information in the executable image, in a section named ".trace". To see which sections are in an image, use the command:

```
link -dump -summary your_app_name.exe
```

To see more detailed information, use the command:

```
link -dump -headers your_app_name.exe
```

On Windows systems, when requesting traceback, you must set Enable Incremental Linking in the VS .NET\* IDE Linker Options to No. On systems using IA-32 architecture and Intel® 64 architecture, you must also set Omit Frame Pointers (the /Oy option) in the Optimization Options to "No."

On Linux systems, to display the section headers in the image (including the header for the .trace section, if any), use the command:

```
objdump -h your app name.exe
```

On Mac OS X systems, to display the section headers in the image, use the command:

```
otool -1 your_app_name.exe
```

#### **Alternate Options**

None

#### tune

Determines the version of the architecture for which the compiler generates instructions.

#### **IDE Equivalent**

#### None

#### **Architectures**

IA-32, Intel® 64 architectures

## **Syntax**

Linux and Mac OS X: -tune keyword
Windows: /tune:keyword

## Arguments

*keyword* Specifies the processor type. Possible values are:

pn1 Optimizes for the Intel® Pentium®

processor.

pn2 Optimizes for the Intel® Pentium® Pro,

Intel® Pentium® II, and Intel®

Pentium® III processors.

pn3 Optimizes for the Intel® Pentium® Pro,

Intel® Pentium® II, and Intel®

Pentium® III processors. This is the

same as specifying pn2.

pn4 Optimizes for the Intel® Pentium® 4

processor.

#### Default

pn4 The compiler optimizes for the Intel® Pentium® 4 processor.

### Description

This option determines the version of the architecture for which the compiler generates instructions.

On systems using Intel® 64 architecture, only *keyword* pn4 is valid.

Intel® Fortran Compiler User and Reference Guides **Alternate Options** None u (Linux\* and Mac OS\* X) See warn. u (Windows\*) Undefines all previously defined preprocessor values. **IDE Equivalent** Windows: Preprocessor > Undefine All Preprocessor Definitions Linux: None Mac OS X: None Architectures IA-32, Intel® 64, IA-64 architectures Syntax Linux and Mac OS X: None Windows: /u Arguments None Default

OFF Defined preprocessor values are in effect until they are undefined.

## Description

This option undefines all previously defined preprocessor values.

To undefine specific preprocessor values, use the /U option.

None

#### See Also

U compiler option

U

Undefines any definition currently in effect for the specified symbol.

## **IDE Equivalent**

Windows: **Preprocessor > Undefine Preprocessor Definitions** 

Linux: None

Mac OS X: Preprocessor > Undefine Preprocessor Definitions

#### Architectures

IA-32, Intel® 64, IA-64 architectures

#### **Syntax**

Linux and Mac OS X: -Uname

Windows: /Uname

## Arguments

*name* Is the name of the symbol to be undefined.

### Default

OFF Symbol definitions are in effect until they are undefined.

## Description

This option undefines any definition currently in effect for the specified symbol. On Windows systems, use the  $/\mathrm{u}$  option to undefine all previously defined preprocessor values.

## **Alternate Options**

Linux and Mac OS X: None
Windows: /undefine:name

#### See Also

u (Windows) compiler option

#### undefine

See <u>U</u>.

## unroll, Qunroll

Tells the compiler the maximum number of times to unroll loops.

## **IDE Equivalent**

Windows: Optimization > Loop Unroll Count

Linux: None

Mac OS X: Optimization > Loop Unroll Count

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -unroll[=n]

Windows: /Qunroll[:n]

## Arguments

*n* Is the maximum number of times a loop can be

unrolled. To disable loop enrolling, specify 0.

On systems using IA-64 architecture, you can only

specify a value of 0.

#### Default

660

-unroll The compiler uses default heuristics when unrolling loops.
or/Qunroll

### Description

This option tells the compiler the maximum number of times to unroll loops. If you do not specify n, the optimizer determines how many times loops can be unrolled.

## **Alternate Options**

Linux and Mac OS X: -funroll-loops

Windows: /unroll

### See Also

Optimizing Applications: Loop Unrolling

## unroll-aggressive, Qunroll-aggressive

Determines whether the compiler uses more aggressive unrolling for certain loops.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## Syntax

Linux and Mac OS X: -unroll-aggressive

-no-unroll-aggressive

Windows: /Qunroll-aggressive

/Qunroll-aggressive-

## **Arguments**

None

## Default

```
-no-unroll-aggressive The compiler uses default heuristics when or/Qunroll- unrolling loops.

aggressive-
```

## Description

This option determines whether the compiler uses more aggressive unrolling for certain loops. The positive form of the option may improve performance.

On IA-32 architecture and Intel® 64 architecture, this option enables aggressive, complete unrolling for loops with small constant trip counts.

On IA-64 architecture, this option enables additional complete unrolling for loops that have multiple exits or outer loops that have a small constant trip count.

## **Alternate Options**

None

## uppercase, Quppercase

See names.

us

See <u>assume</u>.

## use-asm, Quse-asm

Tells the compiler to produce objects through the assembler.

## **IDE Equivalent**

None

#### **Architectures**

```
-use-asm: IA-32 architecture, Intel® 64 architecture, IA-64 architecture
```

/Quse-asm: IA-64 ar	chitecture
Syntax	
Linux and Mac OS X:	-use-asm
	-no-use-asm
Windows:	/Quse-asm
	/Quse-asm-
Arguments	
None	
Default	
-no-use- The co	empiler produces objects directly.
asm	
or/Quse-	
asm-	
Description	
This option tells the c	ompiler to produce objects through the assembler.
Alternate Options	
None	
V	
Specifies that driver to	ool commands should be displayed and executed.
IDE Equivalent	
None	
Architectures	

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -v[file]

Windows: None

### Arguments

file Is the name of a file.

#### Default

OFF No tool commands are shown.

## Description

This option specifies that driver tool commands should be displayed and executed.

If you use this option without specifying a file name, the compiler displays only the version of the compiler.

If you want to display processing information (pass information and source file names), specify option watch:all.

## **Alternate Options**

Linux and Mac OS X: -watch cmd

Windows: /watch:cmd

#### See Also

dryrun compiler option
watch compiler option

V (Linux\* and Mac OS\* X)

See <u>logo</u>

## V (Windows\*)

See bintext.

## vec, Qvec

Enables or disables vectorization and transformations enabled for vectorization.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64 architectures

## **Syntax**

Linux and Mac OS X: -vec

-no-vec

Windows: /Qvec

/Qvec-

## Arguments

None

#### Default

-vec

Vectorization is enabled.

or/Qvec

## Description

This option enables or disables vectorization and transformations enabled for vectorization.

To disable vectorization and transformations enabled for vectorization, specify – no-vec (Linux and Mac OS X) or /Qvec- (Windows).

## **Alternate Options**

None

## vec-guard-write, Qvec-guard-write

Tells the compiler to perform a conditional check in a vectorized loop.

## IDE Equivalent

None

#### Architectures

IA-32, Intel® 64 architectures

## **Syntax**

```
Linux and Mac OS X: -vec-guard-write
```

-no-vec-guard-write

Windows: /Qvec-guard-write

/Qvec-guard-write-

## Arguments

None

#### Default

```
-no-vec-guard-write The compiler uses default heuristics when or/Qvec-guard-write- checking vectorized loops.
```

## Description

This option tells the compiler to perform a conditional check in a vectorized loop.

This checking avoids unnecessary stores and may improve performance.

## **Alternate Options**

None

## vec-report, Qvec-report

Controls the diagnostic information reported by the vectorizer.

## IDE Equivalent

Windows: Compilation Diagnostics > Vectorizer Diagnostic Level

Linux: None

Mac OS X: Diagnostics > Vectorizer Diagnostic Report

#### Architectures

IA-32, Intel® 64 architectures

## **Syntax**

Linux and Mac OS X: -vec-report[n]

Windows: /Qvec-report[n]

#### Arguments

n Is a value denoting which diagnostic messages to

report. Possible values are:

O Tells the vectorizer to report no

diagnostic information.

1 Tells the vectorizer to report on

vectorized loops.

2 Tells the vectorizer to report on

vectorized and non-vectorized

loops.

3 Tells the vectorizer to report on

vectorized and non-vectorized

loops and any proven or

assumed data dependences.

4 Tells the vectorizer to report on

non-vectorized loops.

5 Tells the vectorizer to report on

non-vectorized loops and the

reason why they were not vectorized.

#### Default

-vec- If the vectorizer has been enabled, it reports diagnosticsreport1 on vectorized loops.

0.7 2 0 0 0

report1

### Description

This option controls the diagnostic information reported by the vectorizer. The vectorizer report is sent to stdout.

If you do not specify n, it is the same as specifying -vec-report1 (Linux and Mac OS X) or /Qvec-report1 (Windows).

The vectorizer is enabled when certain compiler options are specified, such as option -ax or -x (Linux and Mac OS X), option /Qax or /Qx (Windows), option -ax or -ax or -ax h SSE or -ax h SSE2 (Linux and Mac OS X), option

/architecture:SSE or /architecture:SSE2 (Windows).

If this option is specified from within the IDE, the report is included in the build log if the Generate Build Logs option is selected.

#### **Alternate Options**

None

## vms

Causes the run-time system to behave like HP\* Fortran on OpenVMS\* Alpha systems and VAX\* systems (VAX FORTRAN\*).

#### **IDE Equivalent**

Windows: Compatibility > Enable VMS Compatibility

Linux: None

Mac OS X: Compatibility > Enable VMS Compatibility

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

#### **Syntax**

Linux and Mac OS X: -vms

-novms

Windows: /vms

/novms

#### Arguments

None

#### Default

novms The run-time system follows default Intel® Fortran behavior.

## Description

This option causes the run-time system to behave like HP\* Fortran on OpenVMS\* Alpha systems and VAX\* systems (VAX FORTRAN\*). It affects the following language features:

- Certain defaults
  - In the absence of other options, vms sets the defaults as check format and check output\_conversion.
- Alignment
  - Option vms does not affect the alignment of fields in records or items in common blocks. For compatibility with HP Fortran on OpenVMS systems, use align norecords to pack fields of records on the next byte boundary.
- Carriage control default

If option vms and option ccdefault default are specified, carriage control defaults to FORTRAN if the file is formatted and the unit is connected to a terminal.

## INCLUDE qualifiers

/LIST and /NOLIST are recognized at the end of the file name in an INCLUDE statement at compile time. If the file name in the INCLUDE statement does not specify the complete path, the path used is the current directory. Note that if vms is not specified, the path used is the directory where the file that contains the INCLUDE statement resides.

#### Quotation mark character

A quotation mark (") character is recognized as starting an octal constant ("0..7) instead of a character literal ("...").

#### Deleted records in relative files

When a record in a relative file is deleted, the first byte of that record is set to a known character (currently '@'). Attempts to read that record later result in ATTACCNON errors. The rest of the record (the whole record, if vms is not specified) is set to nulls for unformatted files and spaces for formatted files.

## ENDFILE records

When an ENDFILE is performed on a sequential unit, an actual 1-byte record containing a Ctrl/Z is written to the file. If vms is not specified, an internal ENDFILE flag is set and the file is truncated. The vms option does not affect ENDFILE on relative files: these files are truncated.

## Implied logical unit numbers

The vms option enables Intel Fortran to recognize certain environment variables at run time for ACCEPT, PRINT, and TYPE statements and for READ and WRITE statements that do not specify a unit number (such as READ (\*,1000)).

## Treatment of blanks in input

The vms option causes the defaults for the keyword BLANK in OPEN statements to become 'NULL' for an explicit OPEN and 'ZERO' for an implicit OPEN of an external or internal file.

- OPEN statement effects
  - Carriage control defaults to FORTRAN if the file is formatted, and the unit is connected to a terminal. Otherwise, carriage control defaults to LIST. The vms option affects the record length for direct access and relative organization files. The buffer size is increased by 1 to accommodate the deleted record character.
- Reading deleted records and ENDFILE records The run-time direct access READ routine checks the first byte of the retrieved record. If this byte is '@' or NULL ("\0"), then an ATTACCNON error is returned. The run-time sequential access READ routine checks to see if the record it just read is one byte long and contains a Ctrl/Z. If this is true, it returns EOF.

## **Alternate Options**

Linux and Mac OS X: None

Windows: /Qvms

### See Also

align compiler option
ccdefault compiler option
check compiler option

W

See keywords none and nogeneral in warn

W0, W1

See warn.

W0, W1

See warn.

Wa

Passes options to the assembler for processing.

## **IDE Equivalent**

None

#### **Architectures**

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -Wa, option1[, option2,...]

Windows: None

## Arguments

option Is an assembler option. This option is not

processed by the driver and is directly passed to

the assembler.

#### Default

OFF No options are passed to the assembler.

## Description

This option passes one or more options to the assembler for processing. If the assembler is not invoked, these options are ignored.

## **Alternate Options**

None

#### warn

Specifies diagnostic messages to be issued by the compiler.

## **IDE Equivalent**

```
Windows: General > Compile Time Diagnostics (/warn:all, /warn:none)
Compilation Diagnostics > Treat Warnings as Errors (/warn: [no]errors)
Compilation Diagnostics > Treat Fortran Standard Warnings as Errors
(/warn:[no]stderrors)
Compilation Diagnostics > Compile Time Diagnostics (/warn:all,
/warn:none)
Compilation Diagnostics > Warn for Undeclared Symbols
(/warn:[no]declarations)
Compilation Diagnostics > Warn for Unused Variables
(/warn:[no]unused)
Compilation Diagnostics > Warn When Removing %LOC
(/warn:[no]ignore loc)
Compilation Diagnostics > Warn When Truncating Source Line
(/warn:[no]truncated_source)
Compilation Diagnostics > Warn for Unaligned Data
(/warn:[no]alignments)
Compilation Diagnostics > Warn for Uncalled Routine
(/warn:[no]uncalled)
Compilation Diagnostics > Suppress Usage Messages (/warn: [no]usage)
Compilation Diagnostics > Check Routine Interfaces
(/warn:[no]interfaces)
Linux: None
Mac OS X: General > Compile Time Diagnostics (-warn all, -warn
none)
Compiler Diagnostics > Warn For Unaligned Data (-warn
[no]alignments)
Compiler Diagnostics > Warn For Undeclared Symbols (-warn
[no]declarations)
Compiler Diagnostics > Treat Warnings as Errors (-warn error)
```

Compiler Diagnostics > Warn When Removing %LOC (-warn

[no]ignore\_loc)

Compiler Diagnostics > Check Routine Interfaces (-warn

[no]interfaces)

Compiler Diagnostics > Treat Fortran Standard Warnings As Errors (-warn

[no]stderrors)

**Compiler Diagnostics > Warn When Truncating Source Line** (-warn

[no]truncated\_source)

Compiler Diagnostics > Warn For Uncalled Routine (-warn [no]uncalled)

Compiler Diagnostics > Warn For Unused Variables (-warn [no]unused)

Compiler Diagnostics > Suppress Usage Messages (-warn [no]usage)

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: -warn [keyword]

-nowarn

Windows: /warn[:keyword]

/nowarn

### Arguments

keyword Specifies the diagnostic messages to be issued. Possible

values are:

none Disables all warning

messages.

[no]alignments Determines whether

warnings occur for data that

is not naturally aligned.

[no]declarations Determines whether

warnings occur for any

undeclared symbols.

[no]errors Determines whether

warnings are changed to

errors.

[no]general Determines whether

warning messages and

informational messages are

issued by the compiler.

warnings occur when %LOC

is stripped from an actual

argument.

[no]interfaces Determines whether the

compiler checks the

interfaces of all

SUBROUTINEs called and

FUNCTIONs invoked in

your compilation against an

external set of interface

blocks.

[no]stderrors Determines whether

warnings about Fortran

standard violations are

changed to errors.

[no]truncated\_source Determines whether

warnings occur when

source exceeds the

maximum column width in

fixed-format files.

[no]uncalled Determines whether

warnings occur when a

statement function is never

called

[no]unused Determines whether

warnings occur for declared

variables that are never

used.

[no]usage Determines whether

warnings occur for

questionable programming

practices.

all Enables all warning

messages.

#### Default

alignments Warnings are issued about data that is not

naturally aligned.

general All information-level and warning-level

messages are enabled.

usage Warnings are issued for questionable

programming practices.

nodeclarations No errors are issued for undeclared symbols.

noerrors Warning-level messages are not changed to

error-level messages.

noignore_loc	No warnings are issued when ${\tt \$LOC}$ is stripped from an argument.
nointerfaces	The compiler does not check interfaces of SUBROUTINEs called and FUNCTIONs invoked in your compilation against an external set of interface blocks.
nostderrors	Warning-level messages about Fortran standards violations are not changed to error-level messages.
notruncated_source	No warnings are issued when source exceeds the maximum column width in fixed-format files.
nouncalled	No warnings are issued when a statement function is not called.
nounused	No warnings are issued for variables that are declared but never used.

## Description

This option specifies the diagnostic messages to be issued by the compiler.

Option	Description
warn none	Disables all warning messages. This is the same as specifying nowarn.
warn noalignments	Disables warnings about data that is not naturally aligned.
warn	Enables error messages about any undeclared symbols.
declarations	This option makes the default data type of a variable
	undefined (IMPLICIT NONE) rather than using the implicit
	Fortran rules.

Option	Description
warn errors	Tells the compiler to change all warning-level messages to error-level messages; this includes warnings about Fortran standards violations.
warn nogeneral	Disables all informational-level and warning-level diagnostic messages.
warn ignore_loc	Enables warnings when %LOC is stripped from an actual argument.
warn interfaces	Tells the compiler to check the interfaces of all SUBROUTINEs called and FUNCTIONs invoked in your compilation against a set of interface blocks stored separately from the source being compiled.  The compiler generates a compile-time message if the interface used to invoke a routine does not match the interface defined in a .mod file external to the source (that is, in a .mod generated by option gen-interfaces as opposed to a .mod file USEd in the source). The compiler looks for these .mods in the current directory or in the directory specified by the include (-I) or -module option.  Tells the compiler to change all warning-level messages about Fortran standards violations to error-level messages. This option sets the std03 option (Fortran 2003 standard). If you want Fortran 95 standards violations to become errors, you must specify options warn stderrors and std95.
warn truncated_source	Enables warnings when a source line exceeds the maximum column width in fixed-format source files. The maximum column width for fixed-format files is 72, 80, or
	maximum column width for fixed-format files is 12, 00, 01

Option	Description
	132, depending on the setting of the extend-source option. The warn truncated_source option has no effect on truncation; lines that exceed the maximum column width are always truncated. This option does not apply to free-format source files.
warn uncalled	Enables warnings when a statement function is never called.
warn unused	Enables warnings for variables that are declared but never used.
warn nousage	Disables warnings about questionable programming practices. Questionable programming practices, although allowed, often are the result of programming errors; for example: a continued character or Hollerith literal whose first part ends before the statement field and appears to end with trailing spaces. Note that the <code>/pad-source</code> option can prevent this error.
warn all	Enables all warning messages. This is the same as specifying warn. This option does not set options warn errors or warn stderrors. To enable all the additional checking to be performed and force the severity of the diagnostic messages to be severe enough to not generate an object file, specify warn all warn errors or warn all warn stderrors.  On Windows systems: In the Property Pages, Custom means that diagnostics will be specified on an individual basis.

# **Alternate Options**

warn none Linux and Mac OS X: -nowarn, -w, -W0, -warn

nogeneral

Windows: /nowarn,/w, /W0, /warn:nogeneral

warn Linux and Mac OS X: -implicitnone, -u

declarations Windows: /4Yd

warn Linux and Mac OS X: None

nodeclarations Windows: /4Nd

warn general Linux and Mac OS X: -W1

Windows: /W1

warn nogeneral Linux and Mac OS X: -W0, -w, -nowarn, -warn none

Windows: /W0, /w, /nowarn, /warn:none

warn stderrors Linux and Mac OS X: -e90, -e95, -e03

Windows: None

warn nousage Linux and Mac OS X: -cm

Windows: /cm

warn all Linux and Mac OS X: -warn

Windows: /warn

#### watch

Tells the compiler to display certain information to the console output window.

### **IDE Equivalent**

None

# **Architectures**

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: -watch [keyword]

-nowatch

Windows: /watch[:keyword]

/nowatch

# Arguments

keyword Determines what information is displayed. Possible values are:

none	Disables cmd and source.
[no]cmd	Determines whether driver tool
	commands are displayed and
	executed.
[no]source	Determines whether the name of the
	file being compiled is displayed.
all	Enables cmd and source.

### Default

nowatch Pass information and source file names are not displayed to the console output window.

### Description

Tells the compiler to display processing information (pass information and source file names) to the console output window.

Option	Description
watch none	Tells the compiler to not display pass information and source file
	names to the console output window. This is the same as
	specifying nowatch.
watch cmd	Tells the compiler to display and execute driver tool commands.
watch	Tells the compiler to display the name of the file being compiled.

0.41	
Option	Description
source	
watch all	Tells the compiler to display pass information and source file
	names to the console output window. This is the same as
	specifying watch with no keyword.
Alternate Optio	ns
watch Linux	and Mac OS X: -v
cmd Windo	ws: None
See Also	
v compiler opti	on
WB	
Turns a comp	ile-time bounds check into a warning.
IDE Equivalent	
None	
Architectures	
IA-32, Intel® 6	64, IA-64 architectures
Syntax	
Linux and Ma	c OS X: -WB
Windows:	/WB
Arguments	
None	
Default	

OFF Compile-time bounds checks are errors.
Description
This option turns a compile-time bounds check into a warning.
Alternate Options
None
what
Tells the compiler to display its detailed version string.
IDE Equivalent
None
Architectures
IA-32, Intel® 64, IA-64 architectures
Syntax
Linux and Mac OS X: -what
Windows: /what
Arguments
None
Default
OFF The version strings are not displayed.
Description
This option tells the compiler to display its detailed version string.
Alternate Options
None

# winapp

Tells the compiler to create a graphics or Fortran Windows application and link against the most commonly used libraries.

## **IDE Equivalent**

Windows: Libraries > Use Common Windows Libraries

Linux: None

Mac OS X: None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

## **Syntax**

Linux and Mac OS X: None

Windows: /winapp

### Arguments

None

#### Default

OFF No graphics or Fortran Windows application is created.

### Description

This option tells the compiler to create a graphics or Fortran Windows application and link against the most commonly used libraries.

# **Alternate Options**

Linux and Mac OS X: None

Windows: /MG

#### Winline

Enables diagnostics about what is inlined and what is not inlined.
IDE Equivalent
None
Architectures
IA-32, Intel® 64, IA-64 architectures
Syntax
Linux and Mac OS X: -Winline
Windows: None
Arguments
None
Default
OFF No diagnostics are produced about what is inlined and what is not inlined.
Description
This option enables diagnostics about what is inlined and what is not inlined. The diagnostics depend on what interprocedural functionality is available.
Alternate Options
None
WI
Passes options to the linker for processing.
IDE Equivalent
None
Architectures

IA-32, Intel® 64, IA-64 architectures

# Syntax

Linux and Mac OS X: -Wl,option1[,option2,...]

Windows: None

### Arguments

option Is a linker option. This option is not processed by

the driver and is directly passed to the linker.

#### Default

OFF No options are passed to the linker.

## Description

This option passes one or more options to the linker for processing. If the linker is not invoked, these options are ignored.

This option is equivalent to specifying option -Qoption, link, options.

### **Alternate Options**

None

### See Also

Qoption compiler option

### Wp

Passes options to the preprocessor.

### **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

### Syntax

Linux and Mac OS X: -Wp,option1[,option2,...]

Windows: None

#### Arguments

option Is a preprocessor option. This option is not

processed by the driver and is directly passed to

the preprocessor.

#### Default

OFF No options are passed to the preprocessor.

## Description

This option passes one or more options to the preprocessor. If the preprocessor is not invoked, these options are ignored.

This option is equivalent to specifying option -Qoption, fpp, options.

# **Alternate Options**

None

# See Also

Qoption compiler option

## x, Qx

Tells the compiler to generate optimized code specialized for the Intel processor that executes your program.

### **IDE Equivalent**

Windows: Code Generation > Intel Processor-Specific Optimization

Optimization > Use Intel(R) Processor Extensions

Linux: None

# Mac OS X: Code Generation > Intel Processor-Specific Optimization

#### **Architectures**

IA-32, Intel® 64 architectures

### Syntax

Linux and Mac OS X: -xprocessor

Windows: /Qxprocessor

#### Arguments

processor Indicates the processor for which code is

generated. Many of the following descriptions refer to Intel® Streaming SIMD Extensions (Intel® SSE) and Supplemental Streaming SIMD Extensions

(Intel® SSSE). Possible values are:

Host Can generate instructions for

the highest instruction set and

processor available on the

compilation host.

SSE4.2 Can generate Intel® SSE4

Efficient Accelerated String and

Text Processing instructions

supported by Intel® Core™ i7

processors. Can generate

Intel® SSE4 Vectorizing

Compiler and Media

Accelerator, Intel® SSSE3,

SSE3, SSE2, and SSE

instructions and it can optimize for the Intel® Core™ processor

family.

Vectorizing Compiler and
Media Accelerator instructions
for Intel processors. Can
generate Intel® SSSE3, SSE3,
SSE2, and SSE instructions
and it can optimize for Intel®
45nm Hi-k next generation
Intel® Core™
microarchitecture. This
replaces value S, which is
deprecated.

SSE3\_ATOM Can generate MOVBE

instructions for Intel processors

and it can optimize for the

Intel® Atom™ processor and

Intel® Centrino® Atom™

Processor Technology.

SSSE3 Can generate Intel® SSSE3,
SSE3, SSE2, and SSE
instructions for Intel processors
and it can optimize for the
Intel® Core™2 Duo processor
family. This replaces value T,
which is deprecated.

SSE3 Can generate Intel® SSE3,
SSE2, and SSE instructions for
Intel processors and it can

optimize for processors based

on Intel® Core™

microarchitecture and Intel

NetBurst® microarchitecture.

This replaces value P, which is

deprecated.

SSE2 Can generate Intel® SSE2 and

SSE instructions for Intel

processors, and it can optimize

for Intel® Pentium® 4

processors, Intel® Pentium® M processors, and Intel® Xeon® processors with Intel® SSE2.

This value is not available on

Mac OS X systems. This replaces value N, which is

deprecated.

#### Default

Windows systems:

/arch:SSE2

For more information on the default values, see Arguments above, option  ${\mathfrak m}$  (Linux and Mac OS X) and

option arch (Windows).

Linux systems: -

msse2

Mac OS X systems

using IA-32

architecture: SSE3

Mac OS X systems

using Intel® 64

architecture: SSSE3

#### Description

This option tells the compiler to generate optimized code specialized for the Intel processor that executes your program. It also enables optimizations in addition to Intel processor-specific optimizations. The specialized code generated by this option may run only on a subset of Intel processors.

This option can enable optimizations depending on the argument specified. For example, it may enable Intel® Streaming SIMD Extensions 4 (Intel® SSE4), Intel® Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), Intel® Streaming SIMD Extensions 2 (Intel® SSE2), or Intel® Streaming SIMD Extensions (Intel® SSE) instructions. The binaries produced by these values will run on Intel processors that support all of the features for the targeted processor. For example, binaries produced with SSE3 will run on an Intel® Core™ 2 Duo processor, because that processor completely supports all of the capabilities of the Intel® Pentium® 4 processor, which the SSE3 value targets. Specifying the SSSE3 value has the potential of using more features and optimizations available to the Intel® Core™ 2 Duo processor.

Do not use *processor* values to create binaries that will execute on a processor that is not compatible with the targeted processor. The resulting program may fail with an illegal instruction exception or display other unexpected behavior. For example, binaries produced with SSE3 may produce code that will not run on Intel® Pentium® III processors or earlier processors that do not support SSE2 instructions.

Compiling the function main() with any of the *processor* values produces binaries that display a fatal run-time error if they are executed on unsupported processors. For more information, see *Optimizing Applications*.

If you specify more than one *processor* value, code is generated for only the highest-performing processor specified. The highest-performing to lowest-performing *processor* values are: SSE4.2, SSE4.1, SSSE3, SSE3, SSE2. Note that *processor* value SSE3 ATOM does not fit within this group.

Compiler options m and arch produce binaries that should run on processors not made by Intel that implement the same capabilities as the corresponding Intel processors.

Previous value O is deprecated and has been replaced by option -msse3 (Linux and Mac OS X) and option /arch: SSE3 (Windows).

Previous values W and K are deprecated. The details on replacements are as follows:

- Mac OS X systems: On these systems, there is no exact replacement for W or K. You can upgrade to the default option -msse3 (IA-32 architecture) or option -msse3 (Intel® 64 architecture).
- Windows and Linux systems: The replacement for W is -msse2 (Linux) or /arch: SSE2 (Windows). There is no exact replacement for K. However, on Windows systems, /QxK is interpreted as /arch: IA32; on Linux systems, -xK is interpreted as -mia32. You can also do one of the following:
- Upgrade to option -msse2 (Linux) or option /arch: SSE2 (Windows). This
  will produce one code path that is specialized for Intel® SSE2. It will not run
  on earlier processors
- Specify the two option combination -mia32 -axSSE2 (Linux) or /arch:IA32 /QaxSSE2 (Windows). This combination will produce an executable that runs on any processor with IA-32 architecture but with an additional specialized Intel® SSE2 code path.

The -x and /Qx options enable additional optimizations not enabled with option -m or option /arch.

#### **Alternate Options**

None

#### See Also

ax, Qax compiler optioncompiler optionarch compiler option

X

Removes standard directories from the include file search path.

### **IDE Equivalent**

Windows: **Preprocessor > Ignore Standard Include Path** (/noinclude)

Linux: None

Mac OS X: **Preprocessor > Ignore Standard Include Path** (/noinclude)

### Architectures

IA-32, Intel® 64, IA-64 architectures

### Syntax

Linux and Mac OS X: -X

Windows: /X

#### Arguments

None

#### Default

OFF Standard directories are in the include file search path.

## Description

This option removes standard directories from the include file search path. It prevents the compiler from searching the default path specified by the FPATH environment variable.

On Linux and Mac OS X systems, specifying -X (or -noinclude) prevents the compiler from searching in /usr/include for files specified in an INCLUDE statement.

You can use this option with the I option to prevent the compiler from searching the default path for include files and direct it to use an alternate path.

This option affects fpp preprocessor behavior and the USE statement.

### **Alternate Options**

Linux and Mac OS X: -nostding

Windows: /noinclude

### See Also

I compiler option

#### Xlinker

Passes a linker option directly to the linker.

# **IDE Equivalent**

None

#### Architectures

IA-32, Intel® 64, IA-64 architectures

### Syntax

Linux and Mac OS X: -Xlinkeroption

Windows: None

### Arguments

option Is a linker option.

#### Default

OFF No options are passed directly to the linker.

### Description

This option passes a linker option directly to the linker.

If -Xlinker -shared is specified, only -shared is passed to the linker and no special work is done to ensure proper linkage for generating a shared object. -

Xlinker just takes whatever arguments are supplied and passes them directly to the linker.

If you want to pass compound options to the linker, for example "-L

\$HOME/lib", you must use one of the following methods:

```
-Xlinker -L -Xlinker $HOME/lib
-Xlinker "-L $HOME/lib"
-Xlinker -L\ $HOME/lib
```

#### **Alternate Options**

None

#### See Also

```
shared compiler option
link compiler option
```

у

See syntax-only.

## g, Zi, Z7

Tells the compiler to generate full debugging information in the object file.

# **IDE Equivalent**

```
Windows: General > Debug Information Format (/z7, /zd, /zi)
```

Linux: None

Mac OS X: General > Generate Debug Information (-g)

#### Architectures

IA-32, Intel® 64, IA-64 architectures

### **Syntax**

Linux and Mac OS X: -q

Windows: /Zi

/Z7

Arguments

None

Default

OFF No debugging information is produced in the object file.

Description

This option tells the compiler to generate symbolic debugging information in the

object file for use by debuggers.

The compiler does not support the generation of debugging information in

assemblable files. If you specify this option, the resulting object file will contain

debugging information, but the assemblable file will not.

This option turns off O2 and makes O0 (Linux and Mac OS X) or Od (Windows)

the default unless 02 (or another 0 option) is explicitly specified in the same

command line.

On Linux systems using Intel® 64 architecture and Linux and Mac OS X systems

using IA-32 architecture, specifying the -q or -00 option sets the -fno-omit-

frame-pointer option.

For more information on Zi and Z7, see keyword full in debug (Windows\*).

**Alternate Options** 

Linux and Mac OS X: None

Windows: /debug:full (or /debug)

See Also

zd compiler option

Zd

This option has been deprecated. Use keyword minimal in debug (Windows\*).

zero, Qzero

696

Initializes to zero all local scalar variables of intrinsic type INTEGER, REAL, COMPLEX, or LOGICAL that are saved but not yet initialized.

## **IDE Equivalent**

Windows: Data > Initialize Local Saved Scalars to Zero

Linux: None

Mac OS X: Data > Initialize Local Saved Scalars to Zero

#### Architectures

IA-32, Intel® 64, IA-64 architectures

#### Syntax

Linux and Mac OS X: -zero

-nozero

Windows: /Qzero

/Qzero-

# Arguments

None

## Default

-nozero Local scalar variables are not initialized to zero.

or

/Qzero-

#### Description

This option initializes to zero all local scalar variables of intrinsic type INTEGER, REAL, COMPLEX, or LOGICAL that are saved but not yet initialized.

Use -save (Linux and Mac OS X) or /Qsave (Windows) on the command line to make all local variables specifically marked as SAVE.

### **Alternate Options**

None

#### See Also

save compiler option

g, Zi, Z7

Tells the compiler to generate full debugging information in the object file.

# **IDE Equivalent**

Windows: General > Debug Information Format (/z7, /zd, /zi)

Linux: None

Mac OS X: **General > Generate Debug Information** (-g)

#### Architectures

IA-32, Intel® 64, IA-64 architectures

# **Syntax**

Linux and Mac OS X: -g

Windows: /Zi

/Z7

#### Arguments

None

## Default

OFF No debugging information is produced in the object file.

# Description

This option tells the compiler to generate symbolic debugging information in the object file for use by debuggers.

The compiler does not support the generation of debugging information in assemblable files. If you specify this option, the resulting object file will contain debugging information, but the assemblable file will not.

This option turns off O2 and makes O0 (Linux and Mac OS X) or Od (Windows) the default unless O2 (or another O option) is explicitly specified in the same command line.

On Linux systems using Intel® 64 architecture and Linux and Mac OS X systems using IA-32 architecture, specifying the -g or -00 option sets the -fno-omit-frame-pointer option.

For more information on Zi and Z7, see keyword full in debug (Windows\*).

#### **Alternate Options**

Linux and Mac OS X: None

Windows: /debug:full (or /debug)

#### See Also

zd compiler option

### ΖI

See keyword none in libdir

### Zp

See keyword rechbyte in align.

#### Zs

See syntax-only.

#### **Quick Reference Guides and Cross References**

#### **Quick Reference Guides and Cross References**

The topic summarizes Intel® Fortran compiler options used on Windows\* OS, Linux\* OS and Mac OS\* X.

If you want to see the summarized Windows\* OS options, see this topic.

If you want to see the summarized Linux\* OS and Mac OS\* X options, see this topic.

#### Windows\* OS Quick Reference Guide and Cross Reference

The table in this section summarizes Intel® Fortran compiler options used on Windows\* OS . Each summary also shows the equivalent compiler options on Linux\* OS and Mac OS\* X.

If you want to see the summarized Linux\* OS and Mac OS\* X options, see this table.

Some compiler options are only available on systems using certain architectures, as indicated by these labels:

Label	Meaning	
i32	The option is available on systems using IA-32 architecture.	
i64em	The option is available on systems using Intel® 64 architecture.	
i64	The option is available on systems using IA-64 architecture.	
If "only" appears in the label, the option is only available on the identified system		
or architecture.		

If no label appears, the option is available on all supported systems and architectures.

For more details on the options, refer to the <u>Alphabetical Compiler Options</u> section.

The Intel® Fortran Compiler includes the Intel® Compiler Option Mapping tool.

This tool lets you find equivalent options by specifying compiler option -mapopts (Linux and Mac OS X) or /Qmap-opts (Windows).

For information on conventions used in this table, see **Conventions**.

## **Quick Reference of Windows\* OS Options**

The following table summarizes all supported Windows\* OS options. It also shows equivalent Linux\* OS and Mac OS\* X options, if any.

Option	Description	Default	Equivalen Linux* OS
/1	Executes at least one iteration of DO loops.	OFF	-1
/4I{2 4 8}	Specifies the default KIND for integer and logical variables; same as the /integer-size option.	/414	-i{2 4 8
/4L{72 80 132}	Treats the statement field of each fixed-form source line as ending in column 72, 80, or 132; same as the /extend-source option.	/4L72	-72, -80
/4Na, /4Ya	Determines where local variables are stored. /4Na is the same as /save. /4Ya is the same as /automatic.	/4Ya	None
/4Naltparam,/4Yaltparam	Determines whether alternate syntax is allowed for PARAMETER statements; same as the /altparam option).	/4Yaltparam	None
/4Nb, /4Yb	Determines whether checking is performed for run-time failures (same as the /check option).	/4Nb	None
/4Nd, /4Yd	Determines whether error	/4Nd	-warn no

Option	Description	Default	Equivalen Linux* OS
	messages are issued for undeclared symbols. /4Nd is the same as /warn:nodeclarations./4Yd is the same as /warn:declarations.		warn dec
/4Nf, /4Yf	Specifies the format for source files. /4Nf is the same as /fixed. /4Yf is the same as /free.	/4Nf	-fixed, -:
/4Nportlib,/4Yportlib	Determines whether the compiler links to the library of portability routines.	/4Yportlib	None
/4Ns, /4Ys	Determines whether the compiler changes warning messages about Fortran standards violations to error messages. /4Ns is the same as /stand:none. /4Ys is the same as /stand:f90.	/4Ns	-stand n
/4R8,/4R16	Specifies the default KIND for real and complex variables. /4R8 is the same as /real-size:64. /4R16 is the same as		-real-si

Option	Description	Default	Equivalen Linux* OS
	/real-size:128.		
/align[:keyword]	Tells the compiler how to align certain data items.	keywords: nocommons nodcommons records nosequence	-align [k
/allow:keyword	Determines how the fpp preprocessor treats Fortran end-of-line comments in preprocessor directive lines.	keyword: fpp_comments	/allow ke
/[no]altparam	Allows alternate syntax (without parentheses) for PARAMETER statements.	/altparam	-[no]alt]
/arch:processor (i32, i64em)	Tells the compiler to generate optimized code specialized for the processor that executes your program; same as option /architecture.	varies; see option description	-arch proprocessor (i32, i64em
/asmattr:keyword	Specifies the contents of an assembly listing file.	/noasmattr	None
/asmfile[:file dir]	Specifies that an assembly listing file should be generated.	/noasmfile	None
/assume:keyword	Tells the compiler to make	keywords:	-assume k

Option	Description	Default	Equivalen Linux* OS
	certain assumptions.	nobscc	
		nobuffered_io	
		nobyterecl	
		nocc_omp	
		nodummy_aliases	
		nominus0	
		noold_boz	
		old_unit_star	
		old_xor	
		protect_constants	
		noprotect_parens	
		norealloc_lhs	
		source_include	
		nostd_mod_proc_name	
		nounderscore	
		nowriteable-strings	
/[no]automatic	Causes all variables to be allocated to the run-time stack; same as the /auto option.	/Qauto-scalar	-[no]aut
/bigobj	Increases the number of sections that an object file can contain.	OFF	None
/bintext:string	Places a text string into the object file (.obj) being generated by the compiler.	OFF	None
/c	Prevents linking.	OFF	-C

Option	Description	Default	Equivalen Linux* OS
/C	Performs checking for all run-time failures; same as the /check:all option.	OFF	-C
/CB	Performs run-time checking on array subscript and character substring expressions; same as the /check:bounds option.	OFF	-CB
/ccdefault: keyword	Specifies the type of carriage control used when a file is displayed at a terminal screen.	/ccdefault:default	-ccdefau
/check[:keyword]	Checks for certain conditions at run time.	/nocheck	-check [k
/cm	Disables all messages about questionable programming practices; same as specifying option /warn:nousage.	OFF	-cm
/compile-only	Causes the compiler to compile to an object file only and not link; same as the /c option.	OFF	None
/Qcomplex-limited- range[-]	Enables the use of basic algebraic expansions of some arithmetic operations	/Qcomplex-limited- range-	-[no-]cotrange

Option	Description	Default	Equivalen
	involving data of type COMPLEX.		
/convert: keyword	Specifies the format of unformatted files containing numeric data.	/convert:native	-convert
/CU	Enables run-time checking for uninitialized variables. This option is the same as /check:uninit and /RTCu.	OFF	-CU
/Dname [=value]	Defines a symbol name that can be associated with an optional value.	/noD	-Dname [=
/[no]d-lines	Compiles debugging statements indicated by the letter D in column 1 of the source code.	/nod-lines	-[no]d-1
/[no]dbglibs	Tells the linker to search for unresolved references in a debug run-time library.	/nodbglibs	None
/debug:keyword	Specifies the type of debugging information generated by the compiler in the object file.	/debug:full (IDE) /debug:none (command line)	-debug ke Note: the L X option tal keyword s
/debug- parameters[:keyword]	Tells the compiler to generate debug information	/nodebug-parameters	-debug-p [keyword]

Option	Description	Default	Equivalen Linux* OS
	for PARAMETERs used in a program.		
/define:name [=value]	Defines a symbol name that can be associated with an optional value; same as the /D <name>[=value] option.</name>	OFF	None
/dll	Specifies that a program should be linked as a dynamic-link (DLL) library.	OFF	None
/double-size:size	Defines the default KIND for DOUBLE PRECISION and DOUBLE COMPLEX variables.	/double-size:64	-double-
/E	Causes the Fortran preprocessor to send output to stdout.	OFF	-E
/EP	Causes the Fortran preprocessor to send output to stdout, omitting #line directives.	OFF	-EP
/error-limit:n	Specifies the maximum number of error-level or fatal-level compiler errors allowed for a file specified on the command line.	/error-limit:30	-error-l

Option	Description	Default	Equivalen Linux* OS
/exe:{file dir}	Specifies the name for a built program or dynamic-link library.	OFF	-0
/extend-source[:size]	Specifies the length of the statement field in a fixed-form source file.	/extend-source:72	-extend-
/extfor:ext	Specifies file extensions to be processed by the compiler as Fortran files.	OFF	None
/extfpp:ext	Specifies file extensions to be recognized as a file to be preprocessed by the Fortran preprocessor.	OFF	None
/extlnk:ext	Specifies file extensions to be passed directly to the linker.	OFF	None
/Fn	Specifies the stack reserve amount for the program.	OFF	None
/f66	Tells the compiler to apply FORTRAN 66 semantics.	OFF	-f66
/f77rtl	Tells the compiler to use the run-time behavior of FORTRAN 77.	OFF	-f77rtl
/Fa[:file dir]	Specifies that an assembly listing file should be generated; same as option	OFF	-S

Option	Description	Default	Equivalen Linux* OS
	/asmfile and /S.		
/FAC, /FAS, /FACS	Specifies the contents of an assembly listing file.  /FAc is the same as the  /asmattr:machine  option. /FAs is the same  as the /asmattr:source  option. /FAcs is the same  as the /asmattr:all  option.	OFF	None
/fast	Maximizes speed across the entire program.	OFF	-fast
/Fefile	Specifies the name for a built program or dynamic-link library; same as the /exe option.	OFF	-0
/FI	Specifies source files are in fixed format; same as the /fixed option.	determined by file suffix	-FI
/[no]fixed	Specifies source files are in fixed format.	determined by file suffix	-[no]fix
/[no]fltconsistency	Enables improved floating-point consistency.	/nofltconsistency	-[no]flt
/Fm[file]	Tells the linker to generate a link map file; same as the /map option.	OFF	None

Option	Description	Default	Equivalen Linux* OS
/Fofile	Specifies the name for an object file; same as the /object option.	OFF	None
/fp:keyword	Controls the semantics of floating-point calculations.	/fp:fast=1	-fp-mode
/[no]fpconstant	Tells the compiler that single-precision constants assigned to double-precision variables should be evaluated in double precision.	/nofpconstant	-[no]fpc
/fpe:n	Specifies floating-point exception handling for the main program at run-time.	/fpe:3	-fpen
<pre>/fpp[n] /fpp[="option"]</pre>	Runs the Fortran preprocessor on source files before compilation.	/nofpp	- fpp[n] - fpp[="o
/fpscomp[:keyword]	Specifies compatibility with Microsoft* Fortran  PowerStation or Intel®  Fortran.	/fpscomp:libs	-fpscomp
/FR	Specifies source files are in free format; same as the /free option.	determined by file suffix	-FR
/[no]free	Specifies source files are in free format.	determined by file suffix	-[no]fre

Option	Description	Default	Equivalen Linux* OS
/G2 (i64 only)	Optimizes application performance for systems using IA-64 architecture.	OFF	None
/G2-p9000 (i64 only)	Optimizes for Dual-Core Intel® Itanium® 2 Processor 9000 series.	ON	-mtune i
/G{5 6 7} (i32, i64em)	Optimizes application performance for systems using IA-32 architecture and Intel® 64 architecture. These options have been deprecated.	/G7	None
/GB	Optimizes for Intel® Pentium® Pro, Pentium® II and Pentium® III processors; same as the /G6 option.	OFF	None
/Ge	Enables stack-checking for all functions. Deprecated.	OFF	None
<pre>/gen- interfaces[:[no]source]</pre>	Tells the compiler to generate an interface block for each routine in a source file.		-gen-inte
/Gm	Tells the compiler to use calling convention CVF; same as the /iface:cvf	OFF	None

Option	Description	Default	Equivalen Linux* OS
/Gs[n]	option.  Disables stack-checking for routines with a specified number of bytes of local variables and compiler temporaries.	/Gs4096	None
/GS[-]	Determines whether the compiler generates code that detects some buffer overruns.	/GS-	-f[no-]s
/Gz	Tells the compiler to use calling convention STDCALL; same as the /iface:stdcall option.	OFF	None
/heap-arrays[:size]	Puts automatic arrays and arrays created for temporary computations on the heap instead of the stack.	/heap-arrays-	-heap-ar:
/help[category]	Displays all available compiler options or a category of compiler options; same as the /? option.	OFF	-help
/homeparams	Tells the compiler to store parameters passed in	OFF	None

Option	Description	Default	Equivalen Linux* OS
	registers to the stack.		
/I:dir	Specifies a directory to add to the include path.	OFF	-Idir
/iface:keyword	Specifies the default calling convention and argument-passing convention for an application.	/iface:default	None
/include	Specifies a directory to add to the include path; same as the /I option.	OFF	None
/inline[:keyword]	Specifies the level of inline function expansion.	OFF	None
/[no]intconstant	Tells the compiler to use FORTRAN 77 semantics to determine the kind parameter for integer constants.	/nointconstant	-[no]int
/integer-size:size	Specifies the default KIND for integer and logical variables.	/integer-size:32	-integer
/LD	Specifies that a program should be linked as a dynamic-link (DLL) library.	OFF	None
/libdir[:keyword]	Controls whether linker options for search libraries are included in object files	/libdir:all	None

Option	Description	Default	Equivaler
	generated by the compiler.		
/libs:keyword	Tells the linker to search for unresolved references in a specific run-time library.	/libs:static	None
/link	Passes options to the linker at compile time.	OFF	None
/[no]logo	Displays the compiler version information.	/logo	-[no]log
/map[:file]	Tells the linker to generate a link map file.	/nomap	None
/MD and /MDd	Tells the linker to search for unresolved references in a multithreaded, debug, dynamic-link run-time library.	OFF	None
/MDs and /MDsd	Tells the linker to search for unresolved references in a single-threaded, dynamic-link run-time library.	OFF	None
/MG	Tells the compiler to create a graphics or Fortran Windows application and link against the most commonly used libraries.	OFF	None
/ML and /MLd	Tells the linker to search for unresolved references in a	•	None

Option	Description	Default	Equivalen Linux* OS
	single-threaded, static run- time library.		
/module:path	Specifies the directory where module files should be placed when created and where they should be searched for.	OFF	-module p
/MP[:n]	Creates multiple processes that can be used to compile large numbers of source files at the same time.		-multipl
/MT and /MTd	Tells the linker to search for unresolved references in a multithreaded, static run- time library.	i64em:	None
/MW	Tells the linker to search for unresolved references in a Fortran QuickWin library; same as /libs:qwin.	OFF	None
/MWs	Tells the linker to search for unresolved references in a Fortran standard graphics library; same as /libs:qwins.	OFF	None
/names:keyword	Specifies how source code identifiers and external	/names:uppercase	-names ke

Option	Description	Default	Equivale Linux* O
	names are interpreted.		
/nbs	Tells the compiler to treat the backslash character (\) as a normal character in character literals; same as the /assume:nobscc option.	/nbs	-nbs
/noinclude	Removes standard directories from the include file search path; same as the /x option.	OFF	None
/O[ n ]	Specifies the code optimization for applications.	/02	-O[ n ]
/Obn	Specifies the level of inline function expansion. $n = 0$ , 1, or 2.		-inline
/object:file	Specifies the name for an object file.	OFF	None
/Od	Disables optimizations.	OFF	-00
/0g	Enables global optimizations.	/0g	None
/Op	Enables improved floating-point consistency.	OFF	-mp
/optimize:n	Affects optimizations	/optimize:3 or	-0 <i>n</i>

Option	Description	Default	Equivalen Linux* OS
	performed by the compiler; $n = 1, 2, 3, \text{ or } 4.$	/optimize:4	
/0s	Enables optimizations that do not increase code size and produces smaller code size than O2.	OFF (unless /01 is specified)	-0s
/Ot	Enables all speed optimizations.	/ot (unless /o1 is specified)	None
/Ox	Same as the /02 option.	/Ox	-02
/Oy[-] (i32 only)	Determines whether EBP is used as a general-purpose register in optimizations.	- 、	-f[no-]or pointer (i32, i64em
/P	Causes the Fortran preprocessor to send output to a file, which is named by default; same as the -preprocess-only option.	OFF	-P
/[no]pad-source	Specifies padding for fixed-form source records.	/nopad-source	-[no]pad
/pdbfile[:file]	Specifies that any debug information generated by the compiler should be saved to a program database file.	/nopdbfile	None
/preprocess-only	Causes the Fortran	/nopreprocess-only	-preproc

Option	Description	Default	Equivalen Linux* OS
	preprocessor to send output to a file.		
/Qansi-alias	Tells the compiler to assume the program adheres to the Fortran Standard type aliasability rules.	/Qansi-alias	-ansi-al
/Qauto	Causes all variables to be allocated on the stack, rather than in local static storage.	/Qauto-scalar	-auto
/Qauto-scalar	Causes allocation of scalar variables of intrinsic types INTEGER, REAL, COMPLEX, and LOGICAL to the run-time stack.	/Qauto-scalar	-auto-sc
/Qautodouble	Makes default real and complex variables 8 bytes long; same as the /real-size:64 option.	OFF	-autodou
/Qax <i>p</i> (i32, i64em)	Tells the compiler to generate multiple, processor-specific autodispatch code paths for Intel processors if there is a performance benefit.	OFF	-axp (i32, i64em

Option	Description	Default	Equivalen Linux* OS
/Qchkstk[-] (i64 only)	Enables stack probing when the stack is dynamically expanded at run-time.	/Qchkstk	None
/Qcommon-args	Tells the compiler that dummy (formal) arguments to procedures share memory locations with other dummy arguments or with COMMON variables that are assigned; same as /assume:dummy_aliases.		-common-
/Qcomplex-limited- range[-]	Enables the use of basic algebraic expansions of some arithmetic operations involving data of type COMPLEX.	/Qcomplex-limited- range-	-[no-]co
/Qcpp	Runs the Fortran preprocessor on source files before compilation; same as the /fpp option.	OFF	-cpp
/Qd-lines[-]	Compiles debugging statements indicated by the letter D in column 1 of the source code; can also be specified as /d-lines.	OFF	-[no]d-l
/Qdiag- <i>type</i> : <i>diag-list</i>	Controls the display of	OFF	-diag- <i>ty</i>

Option	Description	Default	Equivalen Linux* OS
	diagnostic information.		
/Qdiag-dump	Tells the compiler to print all enabled diagnostic messages and stop compilation.	OFF	-diag-du
/Qdiag-enable:sv- include	Tells the Static Verifier to analyze include files and source files when issuing diagnostic message.	OFF	-diag-endinclude
/Qdiag-error-limit:n	Specifies the maximum number of errors allowed before compilation stops.	/Qdiag-error- limit:30	-diag-er
/Qdiag-file[:file]	Causes the results of diagnostic analysis to be output to a file.	OFF	-diag-fi
/Qdiag-file- append[:file]	Causes the results of diagnostic analysis to be appended to a file.	OFF	-diag-fi
/Qdiag-id-numbers[-]	Tells the compiler to display diagnostic messages by using their ID number values.	/Qdiag-id-numbers	-[no-]di
/Qdiag-once:id[,id,]	Tells the compiler to issue one or more diagnostic messages only once	OFF	-diag-on
/Qdps	Specifies that the alternate	/Qdps	-dps

Option	Description	Default	Equivalen Linux* OS
	syntax for PARAMETER statements is allowed; same as the /altparam option.		
/Qdyncom "common1,common2,"	Enables dynamic allocation of common blocks at run time.	OFF	-dyncom "common1
/Qextend-source	This is the same as specifying option /extend-source:132.	OFF	-extend-
/Qfast- transcendentals[-]	Enables the compiler to replace calls to transcendental functions with faster but less precise implementations.	/Qfast- transcendentals	-[no-]fa
/Qfma[-] (i64 only)	Enables the combining of floating-point multiplies and add/subtract operations.	/Qfma	-[no-]fm (i64 only; L
/Qfnalign[:n] (i32, i64em)	Tells the compiler to align functions on an optimal byte boundary.	/Qfnalign-	-falign- (i32, i64em
/Qfnsplit[-] (i32, i64)	Enables function splitting.	/Qfnsplit-	-[no-]fn (i64 only; L
/Qfp-port[-] (i32, i64em)	Rounds floating-point results after floating-point operations.	/Qfp-port-	-[no-]fp (i32, i64em

Option	Description	Default	Equivalen Linux* OS
/Qfp-relaxed[-] (i64 only)	Enables use of faster but slightly less accurate code sequences for math functions, such as divide and sqrt.	/Qfp-relaxed-	-[no-]fp- (i64 only; L
/Qfp-speculation:mode	Tells the compiler the mode in which to speculate on floating-point operations.	/Qfp- speculation:fast	-fp-spec
/Qfp-stack-check (i32, i64em)	Generates extra code after every function call to ensure that the FP (floating-point) stack is in the expected state.	OFF	-fp-staci
/Qfpp[n]	Runs the Fortran preprocessor on source files before compilation.	/nofpp	-fpp[ <i>n</i> ]
/Qftz[-]	Flushes denormal results to zero.	i64: /Qftz- i32, i64em: /Qftz	-[no-]ft
/Qglobal-hoist[-]	Enables certain optimizations that can move memory loads to a point earlier in the program execution than where they appear in the source.	/Qglobal-hoist-	-[no-]gl
/QIA64-fr32 (i64 only)	Disables use of high floating-point registers.	OFF	None

Default

Equivalen

Description

Option

				Linux* OS
	QIfist 32 only)	Enables fast float-to-integer conversions; same as option /Qrcd.	OFF	None
/(	Qinline-debug-info	Produces enhanced source position information for inlined code.	OFF	-inline-
/(	Qinline-dllimport[-]	Determines whether dllimport functions are inlined.	/Qinline-dllimport	None
/(	Qinline-factor==n	Specifies the percentage multiplier that should be applied to all inlining options that define upper limits.	/Qinline-factor-	-inline-
/(	Qinline-forceinline	Specifies that an inline routine should be inlined whenever the compiler can do so.	OFF	-inline-
	Qinline-max-per- ompile= <i>n</i>	Specifies the maximum number of times inlining may be applied to an entire compilation unit.	/Qinline-max-per- compile-	-inline-
	Qinline-max-per- outine=n	Specifies the maximum number of times the inliner may inline into a particular routine.	/Qinline-max-per- routine-	-inline-

Option	Description	Default	Equivalen Linux* OS
/Qinline-max-size=n	Specifies the lower limit for the size of what the inliner considers to be a large routine.	/Qinline-max-size-	-inline-
/Qinline-max-total-size=n	Specifies how much larger a routine can normally grow when inline expansion is performed.	/Qinline-max-total- size-	-inline-
/Qinline-min-size=n	Specifies the upper limit for the size of what the inliner considers to be a small routine.	/Qinline-min-size-	-inline-
/Qinstruction=[no]movbe	Determines whether MOVBE instructions are generated for Intel® processors.	/Qinstruction:movbe	- minstruc
/Qinstrument- functions[-]	Determines whether function entry and exit points are instrumented.	/Qinstrument-functions-	-f[no-]i:
/Qip[-]	Enables additional single- file interprocedural optimizations.	OFF	-[no-]ip
/Qip-no-inlining	Disables full and partial inlining enabled by -ip.	OFF	-ip-no-i:
/Qip-no-pinlining (i32, i64em)	Disables partial inlining.	OFF	-ip-no-p

Option	Description	Default	Equivalen Linux* OS
/QIPF-flt-eval-method0 (i64 only)	Tells the compiler to evaluate the expressions involving floating-point operands in the precision indicated by the variable types declared in the program. Deprecated.	OFF	-[no-]IP: method0 (i64 only; L
/QIPF-fltacc[-] (i64 only)	Tells the compiler to apply optimizations that affect floating-point accuracy.  Deprecated.	/QIPF-fltacc-	-IPF-flta (i64 only; L
/QIPF-fma (i64 only)	Enables the combining of floating-point multiplies and add/subtract operations.  Deprecated; use /Qfma.		-IPF-fma (i64 only; L
/QIPF-fp-relaxed (i64 only)	Enables use of faster but slightly less accurate code sequences for math functions, such as divide and sqrt. Deprecated; use /Qfp-relaxed.	/QIPF-fp-relaxed-	-IPF-fp-:
/Qipo[n]	Enables multifile IP optimizations between files.	OFF	-ipo[n]
/Qipo-c	Generates a multifile object file that can be used in further link steps.	OFF	-ipo-c

Option	Description	Default	Equivalen Linux* OS
/Qipo-jobs: n	Specifies the number of commands to be executed simultaneously during the link phase of Interprocedural Optimization (IPO).	/Qipo-jobs:1	-ipo-joba
/Qipo-S	Generates a multifile assembly file that can be used in further link steps.	OFF	-ipo-S
/Qipo-separate	Generates one object file per source file.	OFF	-ipo-sepa
/Qivdep-parallel (i64 only)	Tells the compiler that there is no loop-carried memory dependency in any loop following an IVDEP directive.	OFF	-ivdep-pa
/Qkeep-static-consts[-]	Tells the compiler to preserve allocation of variables that are not referenced in the source.	/Qkeep-static- consts-	-f[no-]ke
/Qlocation,string,dir	Specifies a directory as the location of the specified tool in string.	OFF	-Qlocati
/Qlowercase	Causes the compiler to ignore case differences in identifiers and to convert	OFF	-lowerca

Option	Description	Default	Equivalen Linux* OS
	external names to lowercase; same as the /names:lowercase option.		
/Qmap-opts	Maps one or more Windows* compiler options to their equivalent on a Linux* system (or vice versa).	OFF	-map-opt;
/Qnobss-init	Places any variables that are explicitly initialized with zeros in the DATA section.	OFF	-no-bss-
/Qonetrip	This is the same as specifying option /onetrip.	OFF	-onetrip
/Qopenmp	Enables the parallelizer to generate multithreaded code based on OpenMP* directives.	OFF	-openmp
/Qopenmp-lib:type	Lets you specify an  OpenMP* run-time library to use for linking.	/Qopenmp-lib:legacy	-openmp- (Linux only)
/Qopenmp-link:library	Lets you specify an  OpenMP* run-time library to use for linking.	/Qopenmp-lib:legacy	-openmp- (Linux only)
/Qopenmp-profile	Enables analysis of	OFF	-openmp-

Option	Description	Default	Equivalen Linux* OS
	OpenMP* applications.		(Linux only)
/Qopenmp-report[n]	Controls the OpenMP parallelizer's level of diagnostic messages.	/Qopenmp-report1	-openmp-:
/Qopenmp-stubs	Enables compilation of OpenMP programs in sequential mode.	OFF	-openmp-
/Qopenmp-	Lets you specify an	/Qopenmp-	-openmp-
threadprivate:type	OpenMP* threadprivate	threadprivate:legacy	type
	implementation.		(Linux only)
/Qopt-block-factor:n	Lets you specify a loop blocking factor.	OFF	-opt-blo
/Qopt-jump-	Enables or disables	/Qopt-jump-	-opt-jum
tables:keyword	generation of jump tables for switch statements.	tables:default	tables=k
/Qopt-loadpair[-]	Enables or disables	/Qopt-loadpair-	-[no-]op
(i64 only)	loadpair optimization.		(i64 only; L
/Qopt-mem-bandwidthn	Enables performance	/Qopt-mem-bandwidth0	-opt-mem
(i64 only)	tuning and heuristics that	for serial compilation;	(i64 only; L
	control memory bandwidth	/Qopt-mem-bandwidth1	
	use among processors.	for parallel compilation	
/Qopt-mod-versioning[-]	Enables or disables	/Qopt-mod-	-[no-]op
(i64 only)	versioning of modulo	versioning-	versioni
	operations for certain types of operands.		(i64 only; L
/Qopt-multi-version-	Tells the compiler to use	/Qopt-multi-version-	-[no-]op

Option	Description	Default	Equivalen Linux* OS
aggressive[-] (i32, i64em)	aggressive multi-versioning to check for pointer aliasing and scalar replacement.	aggressive-	version-a
/Qopt-prefetch[:n]	Enables prefetch insertion optimization.	<pre>i64: /Qopt-prefetch i32, i64em: /Qopt- prefetch-</pre>	-opt-pre:
<pre>/Qopt-prefetch-initial- values[-] (i64 only)</pre>	Enables or disables prefetches that are issued before a loop is entered.	/Qopt-prefetch- initial-values	-[no-]op initial- (i64 only; L
<pre>/Qopt-prefetch-issue- excl-hint[-] (i64 only)</pre>	Determines whether the compiler issues prefetches for stores with exclusive hint.	/Qopt-prefetch- issue-excl-hint-	-[no-]op
/Qopt-prefetch-next- iteration[-][:n] (i64 only)	Enables or disables prefetches for a memory access in the next iteration of a loop.	-opt-prefetch-next- iteration	/Qopt-proiteration (i64 only; L
/Qopt-ra-region- strategy[:keyword] (i32, i64em)	Selects the method that the register allocator uses to partition each routine into regions.	/Qopt-ra-region- strategy:default	-opt-ra-: strategy (i32, i64em
/Qopt-report:n	Tells the compiler to generate an optimization report to $stderr$ .	/Qopt-report:2	-opt-rep
/Qopt-report-file:file	Tells the compiler to generate an optimization	OFF	-opt-rep

Option	Description	Default	Equivalen Linux* OS
	report named file.		
/Qopt-report-help	Displays the optimizer phases available for report generation.	OFF	-opt-rep
/Qopt-report-	Specifies an optimizer	OFF	-opt-rep
phase:phase	phase to use when optimization reports are generated.		phase=ph
/Qopt-report-	Generates a report on all	OFF	-opt-rep
routine:string	routines or the routines containing the specified string.		routine=;
/Qopt-streaming-	Enables generation of	/Qopt-streaming-	-opt-str
stores:keyword	streaming stores for	stores:auto	keyword
(i32, i64em)	optimization.		(i32, i64em
/Qopt-subscript-in-	Determines whether the	/Qopt-subscript-in-	-[no-]op
range[-]	compiler assumes no	range-	in-range
(i32, i64em)	overflows in the		(i32, i64em
	intermediate computation		
	of subscript expressions in loops.		
/Qoption,string,options	Passes options to the	OFF	-Qoption,
	tool specified in string.		
/Qpad	Enables the changing of the variable and array memory layout.	/Qpad-	-pad

Option	Description	Default	Equivalen
			Linux* OS
/Qpad-source	This is the same as specifying option /pad-source.	/Qpad-source-	-pad-sou
/Qpar-adjust-stack:n (i32, i64em)	Tells the compiler to generate code to adjust the stack size for a fiber-based main thread.	/Qpar-adjust-stack:0	) None
/Qpar-report[n]	Controls the diagnostic information reported by the auto-parallelizer.	/Qpar-report1	-par-rep
/Qpar-runtime-control[-	Generates code to perform run-time checks for loops that have symbolic loop bounds.	/Qpar-runtime- control-	-[no-]pa: control
<pre>/Qpar-schedule-keyword [[:] n]</pre>	Specifies a scheduling algorithm for DO loop iterations.	OFF	-par-schekeyword[=
/Qpar-threshold[[:]n]	Sets a threshold for the auto-parallelization of loops.	/Qpar-threshold:100	-par-thr
/Qparallel	Tells the auto-parallelizer to generate multithreaded code for loops that can be safely executed in parallel.	OFF	-paralle
/Qpcn	Enables control of floating-	/Qpc64	-pc <i>n</i>
(i32, i64em)	point significand precision.		(i32, i64em

Option	Description	Default	Equivalen Linux* OS
/Qprec	Improves floating-point precision and consistency.	OFF	-mp1
/Qprec-div[-]	Improves precision of floating-point divides.	/Qprec-div-	-[no-]pr
/Qprec-sqrt	Improves precision of	/Qprec-sqrt-	-prec-sq:
(i32, i64em)	square root implementations.		(i32, i64em
/Qprefetch	Enables prefetch insertion	i64:/Qprefetch	-prefetcl
	<pre>optimization. Deprecated; use /Qopt-prefetch</pre>	i32, i64em: /Qprefetch-	
/Qprof-data-order[-]	Enables or disables data	/Qprof-data-order-	-[no-]pr
	ordering if profiling information is enabled.		(Linux only)
/Qprof-dir dir	Specifies a directory for profiling information output files.	OFF	-prof-di:
/Qprof-file file	Specifies a file name for the profiling summary file.	OFF	-prof-fi
/Qprof-func-order[-]	Enables or disables	/Qprof-func-order-	-[no-]pr
	function ordering if profiling information is enabled.		(Linux only)
/Qprof-gen[:keyword]	Produces an instrumented object file that can be used in profile-guided optimization.	/Qprof-gen-	-prof-ge:
/Qprof-genx	Produces an instrumented	OFF	-prof-ge:

Option	Description	Default	Equivalen Linux* OS
	object file that includes extra source position information. Deprecated; use /Qprof- gen:srcpos.		
/Qprof-hotness- threshold:n	Lets you set the hotness threshold for function grouping and function ordering.	OFF	-prof-ho threshol (Linux only
/Qprof-src-dir[-]	Determines whether directory information of the source file under compilation is considered when looking up profile data records.	/Qprof-src-dir	-[no-]pr
/Qprof-src-root:dir	Lets you use relative directory paths when looking up profile data and specifies a directory as the base.	OFF	-prof-sr
/Qprof-src-root-cwd	Lets you use relative directory paths when looking up profile data and specifies the current working directory as the base.	OFF	-prof-sr
/Qprof-use[:arg]	Enables the use of profiling	OFF	-prof-us

Option	Description	Default	Equivalen Linux* OS
	information during optimization.		
/Qrcd (i32, i64em)	Enables fast float-to-intege conversions.	er OFF	-rcd (i32, i64em
/Qrct (i32, i64em)	Sets the internal FPU rounding control to Truncate.	OFF	-rct (i32, i64em
/Qsafe-cray-ptr	Tells the compiler that Cray* pointers do not alias other variables.	OFF	-safe-cr
/Qsave	Causes variables to be placed in static memory.	/Qauto-scalar	-save
/Qsave-temps[-]	Tells the compiler to save intermediate files created during compilation.	.obj files are saved	-[no-]sa
/Qscalar-rep[-] (i32 only)	Enables scalar replacement performed during loop transformation (requires /03).	/Qscalar-rep-	-[no-]sca (i32 only)
<pre>/Qsfalign[n] (i32 only)</pre>	Specifies stack alignment for functions. n is 8 or 16.	/Qsfalign8	None
/Qsox[-]	Tells the compiler to save the compiler options and version number in the executable.	/Qsox-	-[no-]so:
/Qtcheck	Enables analysis of	OFF	-tcheck

Option	Description	Default	Equivalen
	threaded applications.		(Linux only
/Qtcollect[: lib]	Inserts instrumentation probes calling the Intel(R) Trace Collector API.	OFF	-tcollec (Linux only
/Qtcollect-filter[=file]	Lets you enable or disable the instrumentation of specified functions.	OFF	-tcollec (Linux only
/Qtprofile	Generates instrumentation to analyze multi-threading performance.	OFF	-tprofil (Linux only
/Qtrapuv	Initializes stack local variables to an unusual value.	OFF	-ftrapuv
/Qunroll[:n]	Tells the compiler the maximum number of times to unroll loops; same as the /unroll[:n] option.	/Qunroll	-unroll[=
/Qunroll-aggressive[-] (i32, i64em)	Determines whether the compiler uses more aggressive unrolling for certain loops.	/Qunroll-aggressive-	[no-]un aggressi (i32, i64em
/Quppercase	Causes the compiler to ignore case differences in identifiers and to convert external names to uppercase; same as the	/Quppercase	-upperca

Option	Description	Default	Equivalen
			Linux* OS
	/names:uppercase option.		
/Quse-asm[-] (i64 only)	Tells the compiler to produce objects through the assembler.	/Quse-asm-	-[no-]us
/Quse-msasm-symbols (i32,i64em)	Tells the compiler to use a dollar sign ("\$") when producing symbol names.	OFF	None
/Quse-vcdebug (i32 only)	Tells the compiler to issue debug information compatible with the Visual C++ debugger.	OFF	None
/Qvc7.1 /Qvc8 /Qvc9 (i32, i64em)	Specifies compatibility with Microsoft* Visual C++ or Microsoft* Visual Studio.	varies; see option description	None
/Qvec[-] (i32, i64em)	Enables or disables vectorization and transformations enabled for vectorization.	/Qvec	-[no-]ve
/Qvec-guard-write[-] (i32, i64em)	Tells the compiler to perform a conditional check in a vectorized loop.	/Qvec-guard-write-	-[no-]ve
/Qvec-report[n] (i32, i64em)	Controls the diagnostic information reported by the vectorizer.	/Qvec-report1	-vec-repo

Option	Description	Default	Equivalen Linux* OS
/Qvms	Causes the run-time system to behave like HP Fortran for OpenVMS* Alpha systems and VAX* systems (VAX FORTRAN*) in certain ways; same as the /vms option.	/novms	-vms
/Qxprocessor	Tells the compiler to	varies; see the option	-xproces
(i32, i64em)	generate optimized code specialized for the Intel processor that executes your program.	description	(i32, i64em
/Qzero[-]	Initializes to zero all local scalar variables of intrinsic type INTEGER, REAL, COMPLEX, or LOGICAL that are saved but not yet initialized.	OFF	-[no]zer
/real-size:size	Specifies the default KIND for real variables.	/real-size:32	-real-si
/[no]recursive	Tells the compiler that all routines should be compiled for possible recursive execution.	/norecursive	-[no]rec
/reentrancy: keyword	Tells the compiler to generate reentrant code to support a multithreaded	/noreentrancy	-reentra:

Option	Description	Default	Equivalen Linux* OS
	application.		
/RTCu	Enables run-time checking for uninitialized variables; same as option  /check:uninit.	OFF	-check u
/S	Causes the compiler to compile to an assembly file only and not link.	OFF	-S
/source:file	Tells the compiler to compile the file as a Fortran source file.	OFF	None
/stand:keyword	Tells the compiler to issue compile-time messages for nonstandard language elements.	/nostand	-stand <i>ke</i>
/static	Prevents linking with shared libraries.	/static	-static (Linux only)
/syntax-only	Tells the compiler to check only for correct syntax.	OFF	-syntax-
/Tf file	Tells the compiler to compile the file as a Fortran source file; same as the /source option.	OFF	-Tf file
/[no]threads	Tells the linker to search for unresolved references in a multithreaded run-time		-[no]thr

Option	Description	Default	Equivalen Linux* OS
	library.		
/[no]traceback	Tells the compiler to generate extra information in the object file to provide source file traceback information when a severe error occurs at run time.	/notraceback	-[no]tra
/tune: keyword (i32, i64em)	Determines the version of the architecture for which the compiler generates instructions.	/tune:pn4	-tune <i>key</i> (i32, i64em
/u	Undefines all previously defined preprocessor values.	OFF	None Note: the L X option -u
/Uname	Undefines any definition currently in effect for the specified symbol; same as the /undefine option.	OFF	-Uname
/undefine:name	Undefines any definition currently in effect for the specified symbol; same as option /U.	OFF	None
/unroll[:n]	Tells the compiler the maximum number of times to unroll loops. This is the same as /Qunroll, which	/unroll	-unroll[=

Option	Description	Default	Equivalen Linux* OS
	is the recommended option to use.		
/us	Tells the compiler to append an underscore character to external user-defined names; same as the /assume:underscore option.	OFF	-us
/Vstring	Places the text string specified into the object file (.obj) being generated by the compiler; same as the /bintext option.	OFF	None
/[no]vms	Causes the run-time system to behave like HP* Fortran on OpenVMS* Alpha systems and VAX* systems (VAX FORTRAN*).	/novms	-[no]vms
/w	Disables all warning messages; same as specifying option /warn:none or /warn:nogeneral.	OFF	−W
/Wn	Disables ( n=0) or enables ( n=1) all warning	/W1	-Wn

Option	Description	Default	Equivalen Linux* OS
	messages.		
/warn:keyword	Specifies diagnostic messages to be issued by the compiler.	keywords: alignments general usage nodeclarations noerrors noignore_loc nointerfaces nostderrors notruncated_source nouncalled nounused	-warn key
/watch[:keyword]	Tells the compiler to display certain information to the console output window.	/nowatch	-watch [k
/WB	Turns a compile-time bounds check error into a warning.	OFF	-WB
/what	Tells the compiler to display its detailed version string.	OFF	-what
/winapp	Tells the compiler to create a graphics or Fortran Windows application and link against the most	OFF	None

Option	Description	Default	Equivalen Linux* OS
	commonly used libraries.		
/X	Removes standard directories from the include file search path.	OFF	-X
/Zd	Tells the compiler to generate line numbers and minimal debugging information. Deprecated; use option /debug:minimal.	OFF	None
/Zi or /Z7	Tells the compiler to generate full debugging information in the object file; same as the /debug:full or /debug option.	OFF	-g
/Zl	Prevents any linker search options from being included into the object file; same as the /libdir:none or /nolibdir option.		None
/Zp[n]	Aligns fields of records and components of derived types on the smaller of the size boundary specified or the boundary that will naturally align them; same	/Zp16	-Zp[n]

Option	Description	Default	Equivalen Linux* OS
	as the /align:rec <i>n byte</i> option.		
/Zs	Tells the compiler to perform syntax checking only; same as the /syntax-only option.	OFF	None

#### See Also

-map-opts, /Qmap-opts compiler option

#### Linux\* OS and Mac OS\* X Quick Reference Guide and Cross Reference

The table in this section summarizes Intel® Fortran compiler options used on Linux\* OS and Mac OS\* X. Each summary also shows the equivalent compiler options on Windows\* OS.

If you want to see the summarized Windows\* options, see this table. Some compiler options are only available on systems using certain architectures, as indicated by these labels:

Label	Meaning
i32	The option is available on systems using IA-32 architecture.
i64em	The option is available on systems using Intel® 64 architecture.
i64	The option is available on systems using IA-64 architecture.
If "only" appe	ars in the label, the option is only available on the identified system
or architectur	e.

If no label appears, the option is available on all supported systems and architectures.

For more details on the options, refer to the Alphabetical Compiler Options section.

The Intel® Fortran Compiler includes the Intel® Compiler Option Mapping tool. This tool lets you find equivalent options by specifying compiler option -map-opts (Linux and Mac OS X) or /Qmap-opts (Windows). For information on conventions used in this table, see Conventions.

#### Quick Reference of Linux OS and Mac OS X Options

The following table summarizes all supported Linux OS and Mac OS X options. It also shows equivalent Windows OS options, if any.

Option	Description	Default	Equivalent Option Windows* OS
-1	Executes at least one iteration of DO loops.	OFF	/1
-66	Tells the compiler to use FORTRAN 66 semantics.	OFF	None
-72, -80, -132	Treats the statement field of each fixed-form source line as ending in column 72, 80, or 132; same as the -extend-source option.	-72	/4L{72 80 132}
-align [keyword]	Tells the compiler how to align certain data items.	keywords: nocommons nodcommons records nosequence	/align[:keyword]
-allow	Determines how the	-allow fpp_comments	/allow:[no]fpp_
[no]fpp_comments	fpp preprocessor		

Option	Description	Default	Equivalent Option Windows* OS
	treats Fortran end-of- line comments in preprocessor directive lines.		
-[no]altparam	Allows alternate syntax (without parentheses) for PARAMETER statements.	-altparam	/[no]altparam
-[no-]ansi-alias	Tells the compiler to assume the program adheres to the Fortran Standard type aliasability rules		/Qansi-alias[-]
-arch keyword (i32, i64em)	Tells the compiler to generate optimized code specialized for the processor that executes your program.	varies; see the option description	/arch:keyword (i32, i64em)
-assume keyword	Tells the compiler to make certain assumptions.	keywords: nobscc nobuffered_io nobyterecl nocc_omp nodummy_aliases nominus0	/assume: <i>keyword</i>

Option	Description	Default	Equivalent Option Windows* OS
			Windows" 03
		noold_boz	
		old_unit_star	
		old_xor	
		protect_constants	
		noprotect_parens	
		norealloc_lhs	
		source_include	
		nostd_mod_proc_name	
		underscore	
		no2underscores	
		nowriteable-strings	
-[no]automatic	Causes all variables to be allocated to the run-time stack; same as the -auto option.	-auto-scalar	/[no]automatic
-auto-scalar	Causes allocation of scalar variables of intrinsic types INTEGER, REAL, COMPLEX, and LOGICAL to the run- time stack.	-auto-scalar	/Qauto-scalar
-autodouble	Makes default real and complex variables 8 bytes long; same as the - real-size 64	OFF	/Qautodouble

Option	Description	Default	Equivalent Option Windows* OS
	option.		
-axprocessor (i32, i64em)	Tells the compiler to generate multiple, processor-specific auto-dispatch code paths for Intel processors if there is a performance benefit.	OFF	/Qaxprocessor (i32, i64em)
-Bdir	Specifies a directory that can be used to find include files, libraries, and executables.	OFF	None
-Bdynamic (Linux only)	Enables dynamic linking of libraries at run time.	OFF	None
-Bstatic (Linux only)	Enables static linking of a user's library.	OFF	None
-C	Causes the compiler to compile to an object file only and not link.	OFF	/c
-CB	Performs run-time checks on whether array subscript and	OFF	/CB

Option	Description	Default	Equivalent Option Windows* OS
	substring references are within declared bounds; same as the -check bounds option.		
-ccdefault keyword	Specifies the type of carriage control used when a file is displayed at a terminal screen.	-ccdefault default	/ccdefault: <i>key</i> w
-check [keyword]	Checks for certain conditions at run time.	-nocheck	/check[:keyword]
-cm	Disables all messages about questionable programming practices; same as specifying option - warn nousage.	OFF	/cm
-common-args	Tells the compiler that dummy (formal) arguments to procedures share memory locations with other dummy arguments or with	OFF	/Qcommon-args

Option	Description	Default	Equivalent Option Windows* OS
	COMMON variables that are assigned. This option is the same as -assume dummy_aliases.		
-[no-]complex-limited- range	Enables the use of basic algebraic expansions of some arithmetic operations involving data of type COMPLEX.	-no-complex- limited-range	/Qcomplex-limit range[-]
-convert keyword	Specifies the format of unformatted files containing numeric data.	-convert native	/convert:keywoi
-cpp	Runs the Fortran preprocessor on source files prior to compilation.	OFF	/Qcpp
-CU	Enables run-time checking for uninitialized variables. This option is the same as – check uninit.	OFF	/CU
-cxxlib[=dir]	Tells the compiler to link using the C++	-no-cxxlib	None

Description

Default

**Equivalent Option** 

Option

	·		Windows* OS
	run-time libraries provided by gcc.		
-cxxlib-nostd	Prevents the compiler from linking with the standard C++ library.	-no-cxxlib	None
-Dname [=value]	Defines a symbol name that can be associated with an optional value.	-noD	/Dname [=value]
-[no]d-lines	Compiles debugging statements indicated by the letter D in column 1 of the source code.	-nod-lines	/[no]d-lines <b>0r</b> /
-DD	Compiles debugging statements indicated by the letter D in column 1 of the source code; same as the -d-lines option.	-nod-lines	/d-lines
-debug <i>keyword</i>	Specifies settings that enhance debugging.	-debug none	/debug: keyword  Note: the Windows  different keyword s
-debug-parameters	Tells the compiler to	-nodebug-parameters	/debug-

Option	Description	Default	Equivalent Option Windows* OS
[keyword]	generate debug information for PARAMETERs used in a program.		parameters[:keyw
-diag- <i>type diag-list</i>	Controls the display of diagnostic information.	OFF	/Qdiag-type:di
-diag-dump	Tells the compiler to print all enabled diagnostic messages and stop compilation.	OFF	/Qdiag-dump
-diag-enable sv- include	Tells the Static Verifier to analyze include files and source files when issuing diagnostic message.	OFF	/Qdiag-enable:s
-diag-error-limit n	Specifies the maximum number of errors allowed before compilation stops.	-diag-error-limit 30	/Qdiag-error-li
-diag-file[= file]	Causes the results of diagnostic analysis to be output to a file.	OFF	/Qdiag-file[:f:
<pre>-diag-file-append[= file]</pre>	Causes the results of diagnostic analysis to	OFF	/Qdiag-file- append[:file]

Option	Description	Default	Equivalent Option Windows* OS
	be appended to a file.		
-[no-]diag-id-numbers	Tells the compiler to display diagnostic messages by using their ID number values.	-diag-id-numbers	/Qdiag-id-numbe
-diag-once id[,id,]	Tells the compiler to issue one or more diagnostic messages only once	OFF	/Qdiag-once:id[,
-double-size size	Defines the default KIND for DOUBLE PRECISION and DOUBLE COMPLEX variables.	-double-size 64	/double-size:si
-dps	Specifies that the alternate syntax for PARAMETER statements is allowed; same as the -altparam option.	-dps	/Qdps
-dryrun	Specifies that driver tool commands should be shown but not executed.	OFF	None
-dumpmachine	Displays the target	OFF	None

Option	Description	Default	Equivalent Option Windows* OS
	machine and operating system configuration.		
-dynamic-linkerfile (Linux only)	Specifies a dynamic linker in file other than the default.	OFF	None
-dynamiclib (i32, i64em; Mac OS X only)	Invokes the libtool command to generate dynamic libraries.	OFF	None
-dyncom "common1,common2,"	Enables dynamic allocation of common blocks at run time.	OFF	/Qdyncom "common1,commo
-E	Causes the Fortran preprocessor to send output to stdout.	OFF	/E
-e03, -e95, -e90	Causes the compiler to issue errors instead of warnings for nonstandard Fortran; same as the -warn stderrors option.	OFF	None
-EP	Causes the Fortran preprocessor to send output to stdout,	OFF	/EP

Option	Description	Default	Equivalent Option Windows* OS
	omitting #line directives.		
-error-limit n	Specifies the maximum number of error-level or fatallevel compiler errors allowed for a file specified on the command line; same as -diag-error-limit.	-error-limit 30	/error-limit:n
-extend-source size	Specifies the length of the statement field in a fixed-form source file.	-extend-source 72	/extend-source:
-f66	Tells the compiler to use FORTRAN 66 semantics.	OFF	/f66
-[no]f77rtl	Tells the compiler to use FORTRAN 77 run-time behavior.	OFF	/[no]f77rtl
-f[no-]alias	Determines whether aliasing should be assumed in the program.	ON	None
-falign-functions[=n]	Tells the compiler to	-no-falign-	/Qfnalign[:n]

Option	Description	Default	Equivalent Option Windows* OS
(i32, i64em)	align functions on an optimal byte boundary.	functions	(i32, i64em)
-falign-stack[=mode] (i32 only)	Tells the compiler to align functions on an optimal byte boundary.	-falign- stack=default	None
-fast	Maximizes speed across the entire program.	OFF	/fast
-[no-]fast- transcendentals	Enables the compiler to replace calls to transcendental functions with faster but less precise implementations.	ON	/Qfast-transcer
-fcode-asm	Produces an assembly file with optional machine code annotations.	OFF	/FAc
-f[no-]exceptions	Enables exception handling table generation.	-fno-exceptions	None
-f[no-]fnalias	Specifies that aliasing should be assumed within	OFF	-ffnalias

Option	Description	Default	Equivalent Option Windows* OS
	functions.		
-FI	Specifies source files are in fixed format; same as the -fixed option.	determined by file suffix	/FI
-f[no-]inline	Tells the compiler to inline functions declared with cDEC\$ ATTRIBUTES FORCEINLINE.		None
-f[no-]inline-	Enables function	-finline-functions	/0b2
functions	inlining for single file compilation.		
-finline-limit=n	Lets you specify the maximum size of a function to be inlined.	OFF	None
-f[no-]instrument-	Determines whether	-fno-instrument-	/Qinstrument-fu
functions	function entry and exit points are instrumented.	functions	]
-[no]fixed	Specifies source files are in fixed format.	determined by file suffix	/[no]fixed
-f[no-]keep-static- consts	Tells the compiler to preserve allocation of variables that are not referenced in the	-fno-keep-static- consts	/Qkeep-static-o

Option	Description	Default	Equivalent Option Windows* OS
	source.		
-[no]fltconsistency	Enables improved floating-point consistency.	OFF	/[no]fltconsist
-[no-]fma (i64 only; Linux only)	Enables the combining of floating-point multiplies and add/subtract operations.	-fma	/Qfma[-] (i64 only)
-f[no-]math-errno	Tells the compiler that errno can be reliably tested after calls to standard math library functions.	-fno-math-errno	None
-fminshared	Tells the compiler to treat a compilation unit as a component of a main program and not to link it as a shareable object.	OFF	None
-[no-]fnsplit (i64 only; Linux only)	Enables function splitting.	OFF	/Qfnsplit[-] (i32, i64)
<pre>-f[no-]omit-frame- pointer (i32, i64em)</pre>	Determines whether EBP is used as a general-purpose	-fomit-frame- pointer (unless option -00 or -g is specified)	/Oy[-] (i32 only)

Option	Description	Default	Equivalent Option Windows* OS
	register in optimizations. This is the same as specifying option – fp, which is deprecated.		
-fp-model <i>keyword</i>	Controls the semantics of floating-point calculations.	-fp-model fast	/fp:keyword
-[no-]fp-port	Rounds floating-point results after floating-point operations, so rounding to user-declared precision happens at assignments and type conversions (some impact on speed).	-no-fp-port	/Qfp-port[-]
-[no-]fp-relaxed (i64 only; Linux only)	Enables use of faster but slightly less accurate code sequences for math functions, such as divide and sqrt.	-no-fp-relaxed	/Qfp-relaxed[-] (i64 only)
-fp-speculation=mode	Tells the compiler the mode in which to	-fp- speculation=fast	/Qfp-speculatio

Option	Description	Default	Equivalent Option Windows* OS
	speculate on floating- point operations.		
-fp-stack-check (i32, i64em)	Generates extra code after every function call to ensure that the FP (floating-point) stack is in the expected state.		/Qfp-stack-chec (i32, i64em)
-[no]fpconstant	Tells the compiler that single-precision constants assigned to double-precision variables should be evaluated in double precision.	OFF	/[no]fpconstant
-fpe <i>n</i>	Specifies floating- point exception handling at run time for the main program.	-fpe3	/fpe:n
-f[no-]pic,-fPIC	Generates position- independent code.	-fno-pic	None
-fpie (Linux only)	Tells the compiler to generate position-independent code.	OFF	None
-fpp[n] Or -fpp[="option"]	Runs the Fortran preprocessor on	-nofpp	/fpp[n] <b>or</b> /fpp[="

Option	Description	Default	Equivalent Option Windows* OS
	source files prior to compilation.		
-fpscomp [keyword]	Specifies compatibility with Microsoft* Fortran PowerStation or Intel® Fortran.	-fpscomp libs	/fpscomp[:keywor
-FR	Specifies source files are in free format; same as the -free option.	determined by file suffix	/FR
-fr32 (i64 only; Linux only)	Disables use of high floating-point registers.	OFF	None
-[no]free	Specifies source files are in free format.	determined by file suffix	/[no]free
-fsource-asm	Produces an assembly file with optional source code annotations.	OFF	/FAs
-f[no-]stack-security-check	Determines whether the compiler generates code that detects some buffer overruns; same as - f[no-]stack-	-fno-stack- security-check	/GS[-]

Option	Description	Default	Equivalent Option Windows* OS
	protector.		
-fsyntax-only	Specifies that the source file should be checked only for correct syntax; same as -syntax-only.	OFF	None
-ftrapuv	Initializes stack local variables to an unusual value.	OFF	/Qtrapuv
-[no-]ftz	Flushes denormal	i64: -no-ftz	/Qftz[-]
	results to zero.	i32, i64em: -ftz	
-[no-]func-groups (i32, i64em; Linux only)	Enables or disables function grouping if profiling information is enabled. This option is deprecated, use -prof-func-groups.	-no-func-groups	None
-funroll-loops	Tells the compiler to unroll user loops based on the default optimization heuristics; same as -unroll, which is the recommended option.	-funroll-loops	/Qunroll

Option	Description	Default	Equivalent Option Windows* OS
-fverbose-asm	Produces an assembly file with compiler comments, including options and version information.	-fno-verbose-asm	None
<pre>-fvisibility=keyword -fvisibility- keyword= file</pre>	Specifies the default visibility for global symbols; the 2nd form indicates symbols in a file.	- fvisibility=default	None
-g	Produces symbolic debug information in the object file.	OFF	/Zi, /Z7
-gdwarf2	Enables generation of debug information using the DWARF2 format.	OFF	None
-gen-interfaces [[no]source]	Tells the compiler to generate an interface block for each routine in a source file.	-nogen-interfaces	/gen- interfaces[:[no]
-[no-]global-hoist	Enables certain optimizations that can move memory loads to a point earlier in the program execution than where	-global-hoist	/Qglobal-hoist[

Option	Description	Default	Equivalent Option Windows* OS
	they appear in the source.		
-heap-arrays [size]	Puts automatic arrays and arrays created for temporary computations on the heap instead of the stack.	-no-heap-arrays	/heap-arrays[:s
-help [category]	Displays the list of compiler options.	OFF	/help [category
-Idir	Specifies a directory to add to the include path.	OFF	/Idir
-i-dynamic	Links Intel-provided libraries dynamically. This is a deprecated option; use – shared-intel.	OFF	None
-i-static	Links Intel-provided libraries statically. This is a deprecated option; use – static-intel.	OFF	None
-i{2 4 8}	Specifies the default KIND for integer and logical variables;	-i4	/4I{2 4 8}

Option	Description	Default	Equivalent Option Windows* OS
	same as the - integer-size option.		
-idirafter <i>dir</i>	Adds a directory to the second include file search path.	OFF	None
-implicitnone	Sets the default type of a variable to undefined; same as option warn declarations.	OFF	/4Yd
-inline-debug-info	Produces enhanced source position information for inlined code.		/Qinline-debug-
-inline-factor=n	Specifies the percentage multiplier that should be applied to all inlining options that define upper limits.	-no-inline-factor	/Qinline-factor
-inline-forceinline	Specifies that an inline routine should be inlined whenever the compiler can do so.	OFF	/Qinline-forcei

Option	Description	Default	Equivalent Option Windows* OS
-inline-level=n	Specifies the level of inline function expansion. $n = 0, 1,$ or 2.	-inline-level=2 if - 02 is in effect -inline-level=0 if - 00 is specified	/Obn
-inline-max-per- compile=n	Specifies the maximum number of times inlining may be applied to an entire compilation unit.	-no-inline-max-per- compile	/Qinline-max-pecompile=n
-inline-max-per- routine=n	Specifies the maximum number of times the inliner may inline into a particular routine.	-no-inline-max-per- routine	/Qinline-max-peroutine=n
-inline-max-size=n	Specifies the lower limit for the size of what the inliner considers to be a large routine.	-no-inline-max-size	/Qinline-max-si
-inline-max-total- size=n	Specifies how much larger a routine can normally grow when inline expansion is performed.	-no-inline-max- total-size	/Qinline-max-to
-inline-min-size=n	Specifies the upper limit for the size of what the inliner	-no-inline-min-size	/Qinline-min-si

Option	Description	Default	Equivalent Option Windows* OS
	considers to be a small routine.		
-[no]intconstant	Tells the compiler to use FORTRAN 77 semantics to determine the KIND for integer constants.	-nointconstant	/[no]intconstar
-integer-size size	Specifies the default KIND for integer and logical variables.	-integer-size 32	/integer-size:s
-[no-]ip	Enables additional single-file interprocedural optimizations.	OFF	/Qip[-]
-ip-no-inlining	Disables full and partial inlining enabled by -ip.	OFF	/Qip-no-inlinir
-ip-no-pinlining	Disables partial inlining.	OFF	/Qip-no-pinlini
-IPF-flt-eval-method0 (i64 only; Linux only)	Tells the compiler to evaluate the expressions involving floating-point operands in the precision indicated by the variable types		/QIPF-flt-eval- (i64 only)

Option	Description	Default	Equivalent Option Windows* OS
	declared in the program. Deprecated.		
-IPF-fltacc (i64 only; Linux only)	Tells the compiler to apply optimizations that affect floating-point accuracy.  Deprecated.	-no-IPF-fltacc	/QIPF-fltacc (i64 only)
-IPF-fma (i64 only; Linux only)	Enables the combining of floating-point multiplies and add/subtract operations.  Deprecated; use – fma.	-IPF-fma	/QIPF-fma (i64 only)
-IPF-fp-relaxed (i64 only; Linux only)	Enables use of faster but slightly less accurate code sequences for math functions, such as divide and sqrt.  Deprecated; use - fp-relaxed.	-no-IPF-fp-relaxed	/QIPF-fp-relaxe
-ipo[n]	Enables multifile IP optimizations between files.	OFF	/Qipo[n]
-ipo-c	Generates a multifile	OFF	/Qipo-c

Option	Description	Default	Equivalent Option Windows* OS
	object file that can be used in further link steps.		
-ipo-jobs <i>n</i>	Specifies the number of commands to be executed simultaneously during the link phase of Interprocedural Optimization (IPO).	-ipo-jobs1	/Qipo-jobs:n
-ipo-S	Generates a multifile assembly file that can be used in further link steps.		/Qipo-S
-ipo-separate (Linux only)	Generates one object file per source file.	OFF	/Qipo-separate
-isystem <i>dir</i>	Specifies a directory to add to the start of the system include path.	OFF	None
-ivdep-parallel (i64 only; Linux only)	Tells the compiler that there is no loop-carried memory dependency in any loop following an IVDEP directive.	OFF	/Qivdep-paralle

Option	Description	Default	Equivalent Option Windows* OS
-lstring	Tells the linker to search for a specified library when linking.	OFF	None
-Ldir	Tells the linker where to search for libraries before searching the standard directories.	OFF	None
-[no]logo	Displays compiler version information.	-nologo	/[no]logo
-lowercase	Causes the compiler to ignore case differences in identifiers and to convert external names to lowercase; same as the -names lowercase option.	-lowercase	/Qlowercase
-m[processor] (i32, i64em)	Tells the compiler to generate optimized code specialized for the processor that executes your program.	varies; see option description	/arch
-m32, -m64 (i32, i64em)	Tells the compiler to generate code for IA- 32 architecture or Intel® 64	OFF	None

Option	Description	Default	Equivalent Option Windows* OS
	architecture, respectively.		
-map-opts (Linux only)	Maps one or more Linux* compiler options to their equivalent on a Windows* system (or vice versa).	OFF	/Qmap-opts
-march=processor (i32, i64em; Linux only)	Tells the compiler to generate code for a specified processor.	i32: OFF i64em: - march=pentium4	None
-mcmodel=mem_model (i64em only; Linux only)	Tells the compiler to use a specific memory model to generate code and store data.	-mcmodel=small	None
-mdynamic-no-pic (i32 only; Mac OS X only)	Generates code that is not position-independent but has position-independent external references.	OFF	None
-mieee-fp	Tells the compiler to use IEEE floating point comparisons. This is the same as specifying option – fltconsistency or		/fltconsistency

Option	Description	Default	Equivalent Option Windows* OS
	-mp.		
- minstruction=[no]movbe	are generated for	-minstruction=movbe	/Qinstruction=[
-mixed_str_len_arg	Intel® processors.  Tells the compiler that the hidden length passed for a character argument is to be placed immediately after its corresponding character argument in the argument list.	OFF	/iface:mixed_st
-module path	Specifies the directory where module files should be placed when created and where they should be searched for.	OFF	/module:path
-mp	Enables improved floating-point consistency.	OFF	/Op
-mp1	Improves floating- point precision and consistency.	OFF	/Qprec

Option	Description	Default	Equivalent Option Windows* OS
-m[no-]relax (i64 only)	Determines whether the compiler passes linker option -relax to the linker.	-mno-relax	None
-mtune=processor	Performs optimizations for a particular processormtune=itanium2 is equivalent to /G2.	<pre>i32: -mtune=pentium4 i64: - mtune=itanium2- p9000</pre>	None
-multiple-processes= n	Creates multiple processes that can be used to compile large numbers of source files at the same time.	OFF	/MP: n
-names keyword	Specifies how source code identifiers and external names are interpreted.	-names lowercase	/names:keyword
-nbs	Tells the compiler to treat the backslash character (\) as a normal character in character literals; same as the - assume nobscc option.	-nbs	/nbs

Option	Description	Default	Equivalent Option Windows* OS
-no-bss-init	Tells the compiler to place in the DATA section any variables explicitly initialized with zeros.	OFF	/Qno-bss-init
-nodefaultlibs	Prevents the compiler from using standard libraries when linking.	OFF	None
-nodefine	Specifies that all preprocessor definitions apply only to fpp and not to Intel® Fortran conditional compilation directives.	OFF	/nodefine
-nofor-main	Specifies the main program is not written in Fortran, and prevents the compiler from linking for_main.o into applications.		None
-noinclude	Prevents the compiler from searching in a	OFF	/noinclude

Option	Description	Default	Equivalent Option Windows* OS
	directory previously added to the include path for files specified in an INCLUDE statement.		
-nolib-inline	Disables inline expansion of standard library or intrinsic functions.	OFF	None
-nostartfiles	Prevents the compiler from using standard startup files when linking.	OFF	None
-nostdinc	Removes standard directories from the include file search path; same as the -x option.	OFF	None
-nostdlib	Prevents the compiler from using standard libraries and startup files when linking.	OFF	None
-nus	Disables appending an underscore to external user-defined names; same as the	OFF	None

Option	Description	Default	Equivalent Option Windows* OS
	-assume nounderscore option.		
-ofile	Specifies the name for an output file.	OFF	None
-O[ n ]	Specifies the code optimization for applications.	-02	/O[ n ]
-00	Disables all optimizations.	OFF	/Od
-onetrip	Executes at least one iteration of DO loops.	OFF	/onetrip
-openmp	Enables the parallelizer to generate multithreaded code based on OpenMP* directives.	OFF	/Qopenmp
-openmp-lib type (Linux only)	Lets you specify an OpenMP* run-time library to use for linking.	-openmp-lib legacy	/Qopenmp-lib:ty
-openmp-link <i>library</i>	Controls whether the compiler links to static or dynamic OpenMP run-time	-openmp-link dynamic	/Qopenmp-link:

Option	Description	Default	Equivalent Option Windows* OS
	libraries.		
-openmp-profile (Linux only)	Enables analysis of OpenMP* applications.	OFF	/Qopenmp-profil
-openmp-report[n]	Controls the OpenMP parallelizer's level of diagnostic messages.		/Qopenmp-report
-openmp-stubs	Enables compilation of OpenMP programs in sequential mode.		/Qopenmp-stubs
-openmp-threadprivate	Lets you specify an	-openmp-	/Qopenmp-
type	OpenMP*	threadprivate	threadprivate:t
(Linux only)	threadprivate implementation.	legacy	
-opt-block-factor=n	Lets you specify a loop blocking factor.	OFF	/Qopt-block-fac
-opt-jump-	Enables or disables	-opt-jump-	/Qopt-jump-
tables= <i>keyword</i>	generation of jump tables for switch statements.	tables=default	tables:keyword
-[no-]opt-loadpair (i64 only; Linux only)	Enables or disables loadpair optimization.	-no-opt-loadpair	/Qopt-loadpair  (i64 only)
-opt-malloc-options=n (i32, i64em)	Lets you specify an alternate algorithm for malloc().	-opt-malloc- options=0	None
-opt-mem-bandwidthn	Enables performance	-opt-mem-bandwidth0	/Qopt-mem-bandv

Option	Description	Default	Equivalent Option Windows* OS
(i64 only; Linux only)	tuning and heuristics that control memory bandwidth use among processors.	for serial compilation; - opt-mem-bandwidth1 for parallel compilation	(i64 only)
-[no-]opt-mod- versioning (i64 only; Linux only)	Enables or disables versioning of modulo operations for certain types of operands.		/Qopt-mod-versi
-[no-]opt-multi- version-aggressive (i32, i64em)	Tells the compiler to use aggressive multiversioning to check for pointer aliasing and scalar replacement.	-no-opt-multi- version-aggressive	/Qopt-multi-ver aggressive[-] (i32, i64em)
-opt-prefetch[=n]	Enables prefetch insertion optimization.	<pre>i64: -opt-prefetch i32, i64em: -no-opt- prefetch</pre>	/Qopt-prefetch[
-[no-]opt-prefetch- initial-values (i64 only; Linux only)	Enables or disables prefetches that are issued before a loop is entered.	-opt-prefetch- initial-values	/Qopt-prefetch- values[-] (i64 only)
-[no-]opt-prefetch- issue-excl-hint (i64 only; Linux only)	Determines whether the compiler issues prefetches for stores with exclusive hint.	-no-opt-prefetch- issue-excl-hint	/Qopt-prefetch- excl-hint[-] (i64 only)
-[no-]opt-prefetch-	Enables or disables	-opt-prefetch-next-	/Qopt-prefetch-

			· ·
Option	Description	Default	Equivalent Option Windows* OS
next-iteration (i64 only; Linux only)	prefetches for a memory access in the next iteration of a loop.	iteration	iteration[-][: <i>n</i> ] (i64 only)
-opt-ra-region- strategy[=keyword] (i32, i64em)	Selects the method that the register allocator uses to partition each routine into regions.	-opt-ra-region- strategy=default	/Qopt-ra-region strategy[:keywork(i32, i64em)
-opt-report[n]	Tells the compiler to generate an optimization report to stderr.	-opt-report 2	/Qopt-report[:r
-opt-report-file=file	Specifies the name for an optimization report.	OFF	/Qopt-report-fi
-opt-report-help	Displays the optimizer phases available for report generation.	OFF	/Qopt-report-he
-opt-report- phase=phase	Specifies an optimizer phase to use when optimization reports are generated.	OFF	/Qopt-report-ph
-opt-report-	Tells the compiler to	OFF	/Qopt-report-

Option	Description	Default	Equivalent Option Windows* OS
routine=string	generate reports on the routines containing specified text.		routine:string
-opt-streaming-stores	Enables generation	-opt-streaming-	/Qopt-streaming
keyword	of streaming stores	stores auto	stores:keyword
(i32, i64em)	for optimization.		(i32, i64em)
-[no-]opt-subscript-	Determines whether	-no-opt-subscript-	/Qopt-subscript
in-range	the compiler	in-range	range[-]
(i32, i64em)	assumes no		(i32, i64em)
	overflows in the		
	intermediate		
	computation of		
	subscript expressions		
	in loops.		
-p	Compiles and links	OFF	None
	for function profiling		
	with gprof(1).		
-P	Causes the Fortran	OFF	/P
	preprocessor to send		
	output to a file, which		
	is named by default;		
	same as the -		
	preprocess-only		
	option.		
-[no]pad	Enables the changing	OFF	/Qpad[-]
	of the variable and		

Option	Description	Default	Equivalent Option Windows* OS
	array memory layout.		
-[no]pad-source	Specifies padding for fixed-form source records.	OFF	/[no]pad-source source[-]
-par-report[n]	Controls the diagnostic information reported by the autoparallelizer.	-par-report1	/Qpar-report[n]
-[no-]par-runtime- control	Generates code to perform run-time checks for loops that have symbolic loop bounds.	-no-par-runtime- control	/Qpar-runtime-o
-par-schedule- <i>keyword</i> [=n]	Specifies a scheduling algorithm for DO loop iterations.	OFF	/Qpar-schedule- [[:]n]
-par-threshold[n]	Sets a threshold for the auto- parallelization of loops.	-par-threshold100	/Qpar-threshold
-parallel	Tells the auto- parallelizer to generate multithreaded code	OFF	/Qparallel

Option	Description	Default	Equivalent Option Windows* OS
	for loops that can be safely executed in parallel.		
-pcn (i32, i64em)	Enables control of floating-point significand precision.	-pc80	/Qpcn (i32, i64em)
-ba	Compiles and links for function profiling with gprof(1); same as the -p option.	OFF	None
-pie (Linux only)	Produces a position- independent executable on processors that support it.	OFF	None
-[no-]prec-div	Improves precision of floating-point divides.	-prec-div	/Qprec-div[-]
-[no-]prec-sqrt (i32, i64em)	Improves precision of square root implementations.	-no-prec-sqrt	/Qprec-sqrt[-] (i32, i64em)
-prefetch	Enables prefetch insertion optimization.  Deprecated; use - opt-prefetch.	<pre>i64: -prefetch i32, i64em: -no- prefetch</pre>	/Qprefetch
-preprocess-only	Causes the Fortran	OFF	/preprocess-onl

Option	Description	Default	Equivalent Option Windows* OS
	preprocessor to send output to a file, which is named by default; same as the -P option.		
-print-multi-lib	Prints information about where system libraries should be found.	OFF	None
-[no-]prof-data-order (Linux only)	Enables or disables data ordering if profiling information is enabled.	-no-prof-data-order	/Qprof-data-ord
-prof-dir <i>dir</i>	Specifies a directory for profiling information output files.	OFF	/Qprof-dir dir
-prof-file file	Specifies a file name for the profiling summary file.	OFF	/Qprof-file fil
-[no-]prof-func-groups (i32, i64em; Linux only)	Enables or disables function grouping if profiling information is enabled.	-no-prof-func- groups	None
-[no-]prof-func-order (Linux only)	Enables or disables function ordering if	-no-prof-func-order	/Qprof-func-ord

Option	Description	Default	Equivalent Option Windows* OS
	profiling information is enabled.		
-prof-gen[=keyword]	Produces an instrumented object file that can be used in profile-guided optimization.	-[no-]prof-gen	/Qprof-gen[: <i>key</i> w
-prof-genx	Produces an instrumented object file that includes extra source position information.  Deprecated; use - prof-gen=srcpos.	OFF	/Qprof-genx
-prof-hotness- threshold=n (Linux only)	Lets you set the hotness threshold for function grouping and function ordering.	OFF	/Qprof-hotness-threshold:n
-[no-]prof-src-dir	Determines whether directory information of the source file under compilation is considered when looking up profile data records.	-prof-src-dir	/Qprof-src-dir[
-prof-src-root=dir	Lets you use relative directory paths when	OFF	/Qprof-src-root

Option	Description	Default	Equivalent Option Windows* OS
	looking up profile data and specifies a directory as the base.		
-prof-src-root-cwd	Lets you use relative directory paths when looking up profile data and specifies the current working directory as the base.		/Qprof-src-root
-prof-use[=arg]	Enables the use of profiling information during optimization.	-no-prof-use	/Qprof-use[:arg]
-Qinstall dir	Specifies the root directory where the compiler installation was performed.	OFF	None
-Qlocation,string,dir	Specifies a directory as the location of the specified tool in string.	OFF	/Qlocation,stri
-Qoption,string,options	Passes options to the specified tool in string.	OFF	/Qoption,string
-r8, -r16	Specifies the default KIND for real and complex variables	OFF	/4R8,/4R16

Option	Description	Default	Equivalent Option Windows* OS
	r8 is the same as /real-size:64 r16 is the same as /real-size:128.		
-rcd (i32, i64em)	Enables fast float-to- integer conversions.	OFF	/Qrcd (i32, i64em)
-rct (i32, i64em)	Sets the internal FPU rounding control to Truncate.	OFF	/Qrct (i32, i64em)
-real-size <i>size</i>	Specifies the default KIND for real variables.	-real-size 32	/real-size:size
-recursive	Tells the compiler that all routines should be compiled for possible recursive execution.	-norecursive	/recursive
-reentrancy keyword	Tells the compiler to generate reentrant code to support a multithreaded application.	-noreentrancy	/reentrancy:key
-S	Causes the compiler to compile to an assembly file (.s) only and not link; same as		/S

Option	Description	Default	Equivalent Option Windows* OS
	options /Fa and /asmfile.		
-safe-cray-ptr	Tells the compiler that Cray* pointers do not alias other variables.	OFF	/Qsafe-cray-ptr
-save	Causes variables to be placed in static memory.	-auto-scalar	/Qsave
-[no-]save-temps	Tells the compiler to save intermediate files created during compilation.	-no-save-temps	/Qsave-temps[-]
-[no-]scalar-rep (i32 only)	Enables scalar replacement performed during loop transformation (requires -03).	-no-scalar-rep	/Qscalar-rep[-] (i32 only)
-shared (Linux only)	Tells the compiler to produce a dynamic shared object instead of an executable.	OFF	None
-shared-intel	Links Intel-provided libraries dynamically.	OFF	None
-shared-libgcc (Linux only)	Links the GNU libgcc library dynamically.	-shared-libgcc	None

Option	Description	Default	Equivalent Option Windows* OS
-[no-]sox	Tells the compiler to save the compiler options and version in the executable.	-no-sox	/Qsox[-]
-stand keyword	Causes the compiler to issue compile-time messages for nonstandard language elements.	-nostand	/stand:keyword
-static (Linux only)	Prevents linking with shared libraries.	-static	/static
-staticlib (i32, i64em; Mac OS X only)	Invokes the libtool command to generate static libraries.	OFF	None
-static-intel	Links Intel-provided libraries statically.	OFF	None
-static-libgcc (Linux only)	Links the GNU libgcc library statically.	OFF	None
-std90	Causes the compiler to issue messages for language elements that are not standard in Fortran 90; same as -stand f90.	OFF	/stand:f90

Option	Description	Default	Equivalent Option Windows* OS
-std95	Causes the compiler to issue messages for language elements that are not standard in Fortran 95; same as -stand f95.	OFF	/stand:f95
-std03	Causes the compiler to issue messages for language elements that are not standard in Fortran 2003; same as -std or -stand f03.	OFF	/stand:f03
-syntax-only	Specifies that the source file should be checked only for correct syntax.	OFF	/syntax-only
-T file (Linux only)	Tells the linker to read link commands from the specified file.	OFF	None
-tcheck (Linux only)	Enables analysis of threaded applications.	OFF	/Qtcheck
-tcollect[lib]	Inserts instrumentation	OFF	/Qtcollect[=lib]

Option	Description	Default	Equivalent Option Windows* OS
(Linux only)	probes calling the Intel(R) Trace Collector API.		
<pre>-tcollect-filter[file] (Linux only)</pre>	Lets you enable or disable the instrumentation of specified functions.	OFF	/Qtcollect-filt
-Tf file	Tells the compiler to compile the file as a Fortran source file.	OFF	/Tf file
-[no]threads	Tells the linker to search for unresolved references in a multithreaded run- time library.	i32, i64: -nothreads i64em: -threads	/[no]threads
-tprofile (Linux only)	Generates instrumentation to analyze multi- threading performance.	OFF	/Qtprofile
-[no]traceback	Tells the compiler to generate extra information in the object file to provide source file traceback information when a severe error occurs	-notraceback	/[no]traceback

Option	Description	Default	Equivalent Option Windows* OS
	at run time.		
-tune <i>keyword</i> (i32, i64em)	Determines the version of the architecture for which the compiler generates instructions.	-tune pn4	/tune: keyword (i32, i64em)
-u	Enables error messages about any undeclared symbols; same as the -warn declarations option.	OFF	None Note: the Windows not the same
-Uname	Undefines any definition currently in effect for the specified symbol.	OFF	/Uname
-unroll[n]	Tells the compiler the maximum number of times to unroll loops.  -unroll is the same as option - funroll-loops.		/unroll[:n]
-[no-]unroll- aggressive (i32, i64em)	Determines whether the compiler uses more aggressive unrolling for certain	-no-unroll- aggressive	/Qunroll-aggres

Option	Description	Default	Equivalent Option Windows* OS
	loops.		
-uppercase	Causes the compiler to ignore case differences in identifiers and to convert external names to uppercase; same as the -names uppercase option.	OFF	/Quppercase
-us	Tells the compiler to append an underscore character to external user-defined names; same as the -assume underscore option.		/us
-[no-]use-asm	Tells the compiler to produce objects through the assembler.	-no-use-asm	/Quse-asm[-] (i32 only)
-v[file]	Tells the driver that tool commands should be shown and executed.	OFF	None
-V	Displays the compiler version information; same as the -logo	OFF	None

Option	Description	Default	Equivalent Option Windows* OS
	option.		
-[no-]vec (i32, i64em)	Enables or disables vectorization and transformations enabled for vectorization.	-no-vec	/Qvec[-] (i32, i64em)
-[no-]vec-guard-write (i32, i64em)	Tells the compiler to perform a conditional check in a vectorized loop.	-no-vec-guard-write	/Qvec-guard-wr: (i32, i64em)
-vec-report[n] (i32, i64em)	Controls the diagnostic information reported by the vectorizer.	-vec-report1	/Qvec-report[n] (i32, i64em)
-[no]vms	Causes the run-time system to behave like HP* Fortran on OpenVMS* Alpha systems and VAX* systems (VAX FORTRAN*).	-novms	/[no]vms
-w	Disables all warning messages; same as specifying option - warn none or -warn nogeneral.	OFF	/w

Option	Description	Default	Equivalent Option Windows* OS
-Wn	Disables (n=0) or enables (n=1) all warning messages.	-W1	/Wn
-Wa,o1[,o2,]	Passes options (o1,o2, and so forth) to the assembler for processing.	OFF	None
-warn [keyword]	Specifies diagnostic messages to be issued by the compiler.	keywords: alignments nodeclarations noerrors general noignore_loc nointerfaces nostderrors notruncated_source nouncalled nounused usage	/warn[:keyword]
-[no]watch [keyword]	Tells the compiler to display certain information to the console output window.	-nowatch	/[no]watch[:keyw
-WB	Turns a compile-time bounds check error	OFF	/WB

Option	Description	Default	Equivalent Option Windows* OS
	into a warning.		
-what	Tells the compiler to display its detailed version string.	OFF	/what
-Winline	Enables diagnostics about what is inlined and what is not inlined.	OFF	None
-Wl,option1[,option2,]	Passes options (o1, o2, and so forth) to the linker for processing.	OFF	None
-Wp,option1[,option2,]	Passes options (o1, o2, and so forth) to the preprocessor.	OFF	None
-xp (i32, i64em)	Tells the compiler to generate optimized code specialized for the Intel processor that executes your program.	varies; see the option description	/Qxp (i32, i64em)
-X	Removes standard directories from the include file search path.	OFF	/X
-Xlinker option	Passes a linker	OFF	None

Option	Description	Default	Equivalent Option Windows* OS
	option directly to the linker		
-у	Specifies that the source file should be checked only for correct syntax; same as the -syntax-only option.	OFF	/Zs
-[no]zero	Initializes to zero all local scalar variables of intrinsic type INTEGER, REAL, COMPLEX, or LOGICAL that are saved but not yet initialized.	-nozero	/Qzero[-]
-Zp[n]	Aligns fields of records and components of derived types on the smaller of the size boundary specified or the boundary that will naturally align them.	-Zp16	/Zp[n]

### See Also

-map-opts, /Qmap-opts compiler option

# **Related Options**

# **Related Options**

This topic lists related options that can be used under certain conditions.

# Cluster OpenMP\* Options (Linux\* OS only)

The Cluster OpenMP\* (CLOMP or Cluster OMP) options are available if you have a separate license for the Cluster OpenMP product.

These options can be used on Linux\* operating systems running on Intel® 64 and IA-64 architectures.

Option	Description
-[no-]cluster-openmp	Lets you run an OpenMP program on a cluster.
-[no-]cluster-openmp-profile	Links a Cluster OMP program with profiling information.
-[no-]clomp-sharable- propagation	Reports variables that need to be made sharable by the user with Cluster OpenMP.
-[no-]clomp-sharable-info	Reports variables that the compiler automatically makes sharable for Cluster OpenMP.
-[no-]clomp-sharable-commons	Makes all COMMONs sharable by default for Cluster OpenMP.
-[no-]clomp-sharable-modvars	Makes all variables in modules sharable by default for Cluster OpenMP.
-[no-]clomp-sharable- localsaves	Makes all SAVE variables sharable by default for Cluster OpenMP.
-[no-]clomp-sharable-argexprs	Makes all expressions in function and subroutine call statements sharable by default for Cluster OpenMP.

For more information on these options, see the Cluster OpenMP documentation.

## **Linking Tools and Options**

This topic describes how to use the Intel® linking tools, xild (Linux\* OS and Mac OS\* X) or xilink (Windows\* OS).

The Intel linking tools behave differently on different platforms. The following sections summarizes the primary differences between the linking behaviors.

## **Linux and Mac OS Linking Behavior Summary**

The linking tool invokes the compiler to perform IPO if objects containing IR (intermediate representation) are found. (These are mock objects.) It invokes GNU 1d to link the application.

The command-line syntax for xild is the same as that of the GNU linker: xild [<options>] <normal command-line> where:

- [<options>]: (optional) one or more options supported only by xild.
- <normal command-line>: linker command line containing a set of valid arguments for ld.

To create app using IPO, use the option -ofile as shown in the following example:

```
xild -qipo_fas -oapp a.o b.o c.o
```

The linking tool calls the compiler to perform IPO for objects containing IR and creates a new list of object(s) to be linked. The linker then calls 1d to link the object files that are specified in the new list and produce the application with the name specified by the -o option. The linker supports the -ipo, -ipoN, and -ipo-separate options.

## **Windows Linking Behavior Summary**

The linking tool invokes the Intel compiler to perform multi-file IPO if objects containing IR (intermediate representation) is found. These are mock objects. It invokes Microsoft\* <code>link.exe</code> to link the application.

## **Windows Linking Behavior Summary**

The command-line syntax for the Intel® linker is the same as that of the Microsoft linker:

xilink [<options>] <normal command-line>
where:

- [<options>]: (optional) one or more options supported only by xilink.
- <normal command-line>: linker command line containing a set of valid arguments for the Microsoft linker.

To place the multifile IPO executable in <code>ipo\_file.exe</code>, use the linker option <code>/out:file</code>; for example:

The linker calls the compiler to perform IPO for objects containing IR and creates a new list of object(s) to be linked. The linker calls Microsoft <code>link.exe</code> to link the object files that are specified in the new list and produce the application with the name specified by the <code>/out:file</code> linker option.

#### **Using the Linking Tools**

You must use the Intel linking tools to link your application if the following conditions apply:

- Your source files were compiled with multifile IPO enabled. Multi-file IPO is enabled by specifying the -ipo (Linux and Mac OS X) or /Qipo (Windows) command-line option.
- You normally would invoke either the GNU linker (1d) or the Microsoft linker
   (link.exe) to link your application.

The following table lists the available, case-insensitive options supported by the Intel linking tools and briefly describes the behavior of each option:

<b>Linking Tools Option</b>	Description
-qhelp	Lists the available linking tool options.
	Same as passing no option.

Linking Tools Option	Description
-qnoipo	Disables multi-file IPO compilation.
-qipo_fa[{file dir/}]	Produces assembly listing for the multi- file IPO compilation. You may specify an
	optional name for the listing file, or a
	directory (with the backslash) in which to place the file.
	The default listing name is depends on
	the platform:
	• Linux and Mac OS X: ipo_out.s
	• Windows: ipo_out.asm
	If the Intel linking tool invocation results
	in multi-object compilation, either
	because the application is big or
	because the user explicitly instructed
	the compiler to generate multiple
	objects, the first $.s$ (Linux and Mac OS
	X) or .asm (Windows) file takes its name
	from the -qipo_fa option.
	The compiler derives the names of
	subsequent $.s$ (Linux and Mac OS X)
	or .asm (Windows) files by appending
	an incrementing number to the name,
	for example, foo.asm and foo1.asm
	for ipo_fafoo.asm. The same is true
	for the -qipo_fo option (listed below).
-qipo_fo[{file dir/}]	Produces object file for the multi-file IPO
	compilation. You may specify an
	optional name for the object file, or a

Linking Tools Option	Description
	directory (with the backslash) in which to
	place the file. The default object file
	name is depends on the platform:
	• Linux and Mac OS X: ipo_out.o
	• Windows: ipo_out.obj
-qipo_fas	Add source lines to assembly listing.
-qipo_fac	Adds code bytes to the assembly listing.
-qipo_facs	Add code bytes and source lines to
	assembly listing.
-quseenv	Disables override of existing PATH, LIB,
	and INCLUDE variables.
-lib	Invokes librarian instead of linker.
-libtool	Mac OS X: Invokes libtool to create
	a library instead of 1d.
-dv	Displays version information.

### See Also

Optimizing Applications: Using IPO

# **Fortran Preprocessor Options**

The Fortran preprocessor (fpp) may be invoked automatically or by specifying option fpp.

The following options are available if fpp is in effect.

fpp Option	Description
-В	Specifies that C++-style comments
	should not be recognized.

fpp Option	Description
-C	Specifies that C-style comments should not be recognized. This is the same as specifying -c_com=no.
-c_com={yes no}	Determines whether C-style comments are recognized. If you specify – c_com=no or -C, C-style comments are not recognized. By default, C-style comments are recognized; that is, – c_com=yes.
-Dname	Defines the preprocessor variable name as 1 (one). This is the same as if a – Dname=1 option appeared on the fpp command line, or as if a
	#define name 1 line appeared in the source file processed by fpp.
-Dname=def	Defines name as if by a #define directive. This is the same as if a
	#define name def line appeared in the source file processed by fpp. The -D option has lower precedence than the -U option. That is, if the same name is used in both a -U option and a -D option, the name will be undefined regardless of the order
	of the options.

fpp Option	Description
-e	Tells the compiler to accept extended source lines. For fixed format, lines can contain up to 132 characters. For free format, lines can contain up to 32768 characters.
-e80	Tells the compiler to accept extended source lines. For fixed format, lines can contain up to 80 characters.
-fixed	Tells the compiler to assume fixed format in the source file.
-free	Tells the compiler to assume free format in the source file.
-f_com={yes no}	Determines whether Fortran-style end- of-line comments are recognized or ignored by fpp. If you specify - f_com=no, Fortran style end-of-line comments are processed as part of the preprocessor directive. By default, Fortran style end-of-line comments are recognized by fpp on preprocessor lines and are ignored by fpp; that is, - f_com=yes. For example:  #define max 100 ! max number do i = 1, max + 1  If you specify -f_com=yes, fpp will output
	do $i = 1$ , $100 + 1$

fpp Option	Description
	If you specify -f_com=no, fpp will output
	do i = 1, 100 ! max number + 1
-help	Displays information about fpp options.
-I <dir></dir>	Inserts directory <dir> into the search path for #include files with names not beginning with "/". The <dir> is inserted ahead of the standard list of "include" directories so that #include files with names enclosed in double-quotes (") are searched for first in the directory of the file with the #include line, then in directories named with -I options, and lastly, in directories from the standard list. For #include files with names enclosed in angle-brackets (&lt;&gt;), the directory of the file with the #include line is not searched.</dir></dir>
-m	Expands macros everywhere. This is the same as -macro=yes.
macro={yes no_com no}	Determines the bahavior of macro expansion. If you specify –
	macro=no_com, macro expansion is turned off in comments. If you specify - macro=no, no macro expansion occurs anywhere. By default, macros are expanded everywhere; that is, -

fpp Option	Description
	macro=yes.
-noB	Specifies that C++-style comments should be recognized.
-noC	Specifies that C-style comments should be recognized. This is the same as - c_com=yes.
-noJ	Specifies that F90-style comments should be recognized in a #define line.  This is the same as -f_com=no.
-no-fort-cont	Specifies that IDL style format should be recognized. This option is only for the IDE. Note that macro arguments in IDL may have a C-like continuation character "\" which is different from the Fortran continuation character "&". Fpp should recognize the C-like continuation character and process some other non-Fortran tokens so that the IDL processor can recognize them.
-P	Tells the compiler that line numbering directives should not be added to the output file. This line-numbering directive appears as
-Uname	#line-number file-name  Removes any initial definition of name, where name is an fpp variable that is

fpp Option	Description
	predefined on a particular preprocessor.  Here is a partial list of symbols that may be predefined, depending upon the architecture of the system:  Operating System:APPLE,unix, andlinux  Hardware:i386,ia64,x86_64
-undef	Removes initial definitions for all predefined symbols.
-V	Displays the fpp version number.
-w[0]	Prevents warnings from being output.  By default, warnings are output to stderr.
-Xu	Converts uppercase letters to lowercase, except within characterstring constants. The default is to leave the case as is.
-Xw	Tells the compiler that in fixed-format source files, the blank or space symbol " " is insignificant. By default, the space symbol is the delimiter of tokens for this format.
-Y <dir></dir>	Adds directory <dir> to the end of the system include paths.</dir>

For details on how to specify these options on the compiler command line, see <a href="fpp">fpp</a>, <a href="Qfpp">Qfpp</a>.

# Floating-point Operations

**Overview: Floating-point Operations** 

This section introduces the floating-point support in the Intel® Fortran Compiler and provides information about using floating-point operations in your applications. The section also briefly describes the IEEE\* Floating-Point Standard (IEEE 754).

The following table lists some possible starting points:

If you are trying to	Then start with
Understand the programming objectives	Programming Objectives of Floating-
of floating-point applications	Point Applications
Use the -fp-model (Linux* and Mac	Using the -fp-model or /fp Option
$OS^* X)$ or $/fp$ (Windows*) option	
Set the flush-to-zero (FTZ) or	Setting the FTZ and DAZ Flags
denormals-are-zero (DAZ) flags	
Handle floating-point exceptions	Handling Floating-Point Exceptions
Tuning the performance of floating-point	Overview: Tuning Performance of
applications	Floating-Point Applications
Learn about the IEEE Floating-Point	Overview: Understanding IEEE Floating-
Standard	Point Standard

#### **Floating-point Options Quick Reference**

The Intel® Compiler provides various options for you to optimize floating-point calculations with varying degrees of accuracy and predictability on different Intel architectures. This topic lists these compiler options and provides information about their supported architectures and operating systems.

IA-32, Intel® 64, and IA-64 architectures

Linux* and Mac OS* X	Windows*	Description
-fp-model	/fp	Specifies semantics used
		in floating-point

Linux* and Mac OS* X	Windows*	Description
		<pre>calculations. Values are precise, fast [=1/2], strict, source, double, extended, [no- ]except and except[-]. • _fp-model compiler    option</pre>
-fp-speculation	/Qfp-speculation	Specifies the speculation mode for floating-point operations. Values are fast, safe, strict, and off.  • <u>-fp-speculation</u> compiler option
-prec-div	/Qprec-div	Attempts to use slower but more accurate implementation of floating-point divide. Use this option to disable the divide optimizations in cases where it is important to maintain the full range and precision for floating-point division. Using this option results in greater accuracy with some loss of performance.  Specifying -no-prec-div

Linux* and Mac OS* X	Windows*	Description
		(Linux and Mac OS X) or  /Qprec-div- (Windows)  enables optimizations that result in slightly less precise results than full IEEE division.  -prec-div compiler option
-complex-limited-range	/Qcomplex-limited-range	Enables the use of basic algebraic expansions of some arithmetic operations involving data of type COMPLEX. This can cause performance improvements in programs that use a lot of COMPLEX arithmetic. Values at the extremes of the exponent range might not compute correctly.  -complex-limited-range compiler option
-ftz	/Qftz	The default behavior depends on the architecture. Refer to the following topic for details:  -ftz compiler option
-fpe	/fpe	By default, the Fortran

Linux* and Mac OS* X	Windows*	Description
		compiler disables all
		floating-point exceptions
		and floating underflow is
		gradual.
		This option controls which
		exceptions are enabled by
		the compiler. It also
		controls whether floating-
		point underflow is gradual
		or abrupt.
		• <u>-fpe</u> compiler option

# IA-32 and Intel® 64 architectures

Linux* and Mac OS* X	Windows*	Description	
-prec-sqrt	/Qprec-sqrt	Improves the accuracy of square root implementations, but usin this option may impact	
		<ul><li>speed.</li><li><u>-prec-sqrt</u> compiler option</li></ul>	
-pc	/Qpc	Changes the floating point significand precision. Use this option when compiling applications.  The application must use PROGRAM as the entry point, and you must	

Linux* and Mac OS* X	Windows*	Description
		compile the source file containing PROGRAM with this option.  -pc compiler option
-rcd	/Qrcd	Disables rounding mode changes for floating-point-to-integer conversions.  -rcd compiler option
-fp-port	/Qfp-port	Causes floating-point values to be rounded to the source precision at assignments and casts.  -fp-port compiler option
-mp1	/Qprec	This option rounds floating- point values to the precision specified in the source program prior to comparisons. It also implies -prec-div and - prec-sqrt (Linux and Mac OS X) or /Qprec- div and /Qprec-sqrt (Windows).  -mp1 compiler option

# IA-64 architecture only

Linux*	Windows*	Description

Linux*	Windows*	Description
-IPF-fma	/QIPF-fma	Enables or disables the contraction of floating-point multiply and add/subtract operations into a single operation.  • <u>-IPF-fma</u> compiler option
-IPF-fp-relaxed (deprecated)	/QIPF-fp-relaxed (deprecated)	Enables use of faster but slightly less accurate code sequences for math functions, such as the sqrt() function and the divide operation. As compared to strict IEEE* precision, using this option slightly reduces the accuracy of floating-point calculations performed by these functions, usually limited to the least significant binary digit. This option is deprecated.  • _IPF-fp-relaxed compiler option

# **Understanding Floating-point Operations**

# Using the -fp-model (/fp) Option

The -fp-model (Linux\* and Mac OS\* X) or /fp (Windows\*) option allows you to control the optimizations on floating-point data. You can use this option to tune

the performance, level of accuracy, or result consistency across platforms for floating-point applications.

For applications that do not require support for denormalized numbers, the -fp-model or /fp option can be combined with the -ftz (Linux\*and Mac OS\* X) or /Qftz (Windows\*) option to flush denormalized results to zero in order to obtain improved runtime performance on processors based on:

- IA-32 and Intel® 64 architectures
- IA-64 architecture

You can use keywords to specify the semantics to be used. Possible values of the keywords are as follows:

Keyword	Description
precise	Enables value-safe optimizations on floating-point data and rounds intermediate results to source-defined precision.
fast[=1 2]	Enables more aggressive optimizations on floating-point data.
strict	Enables precise and except, disables contractions, and enables the property that allows modification of the floating-point environment.
source	Enables value-safe optimizations on floating-point data and rounds intermediate results to source-defined precision (same as precise keyword).
double	Rounds intermediate results to 53-bit (double) precision and enables value-safe optimizations.
extended	Rounds intermediate results to 64-bit (extended) precision and enables value-safe optimizations.
<pre>[no-]except (Linux* and Mac OS* X) or</pre>	Determines whether floating-point exception semantics are used.

## Keyword

### **Description**

except[-]

(Windows\*)

The default value of the option is -fp-model fast=1 or /fp:fast=1, which means that the compiler uses more aggressive optimizations on floating-point calculations.



Using the default option keyword -fp-model fast or /fp:fast, you may get significant differences in your result depending on whether the compiler uses x87 or SSE2 instructions to implement floating-point operations. Results are more consistent when the other option keywords are used.

Several examples are provided to illustrate the usage of the keywords. These examples show:

- A small example of source code
   Note that the same source code is considered in all the included examples.
- The semantics that are used to interpret floating-point calculations in the source code
- One or more possible ways the compiler may interpret the source code
   Note that there are several ways the compiler may interpret the code; we show just some of these possibilities.

-fp-model fast or /fp:fast

Example source code:

```
REAL T0, T1, T2;
...
T0 = 4.0E + 0.1E + T1 + T2;
```

When this option is specified, the compiler applies the following semantics:

- Additions may be performed in any order
- Intermediate expressions may use single, double, or extended double precision

 The constant addition may be pre-computed, assuming the default rounding mode

Using these semantics, the following shows some possible ways the compiler may interpret the original code:

```
REAL T0, T1, T2;
...

T0 = (T1 + T2) + 4.1E;
REAL T0, T1, T2;
...

T0 = (T1 + 4.1E) + T2;
```

#### -fp-model source or /fp:source

This setting is equivalent to -fp-model precise or /fp:precise on systems based on the Intel® 64 architecture.

Example source code:

```
REAL T0, T1, T2;
...
T0 = 4.0E + 0.1E + T1 + T2;
```

When this option is specified, the compiler applies the following semantics:

- Additions are performed in program order, taking into account any parentheses
- Intermediate expressions use the precision specified in the source code
- The constant addition may be pre-computed, assuming the default rounding mode

Using these semantics, the following shows a possible way the compiler may interpret the original code:

```
REAL T0, T1, T2;
...
T0 = ((4.1E + T1) + T2);
```

-fp-model strict or /fp:strict

Example source code:

```
REAL T0, T1, T2;
...
T0 = 4.0E + 0.1E + T1 + T2;
```

When this option is specified, the compiler applies the following semantics:

Additions are performed in program order, taking into account any parentheses

- Intermediate expressions use the precision specified in the source code. In Fortran, intermediate expressions always use source precision in modes other than fast.
- The constant addition will not be pre-computed, because there is no way to tell what rounding mode will be active when the program runs.

Using these semantics, the following shows a possible way the compiler may interpret the original code:

```
REAL T0, T1, T2;
...
T0 = REAL ((((REAL)4.0E + (REAL)0.1E) + (REAL)T1) + (REAL)T2);
```

#### See Also

<u>-fp-model compiler option</u> Controls the semantics of floating-point calculations.

/fp compiler option Controls the semantics of floating-point calculations.

### **Floating-point Optimizations**

Application performance is an important goal of the Intel® Compilers, even at default optimization levels. A number of optimizations involve transformations, such as evaluation of constant expressions at compiler time, hoisting invariant expressions out of loops, or changes in the order of evaluation of expressions. These optimizations usually help the compiler produce most efficient code possible. However, this may not be true for floating-point applications, because some optimizations may affect accuracy, reproducibility, and performance. Some optimizations are not consistent with strict interpretation of the ANSI or ISO standards for Fortran, which can result in differences in rounding and small variants in floating-point results that may be more or less accurate than the ANSI-conformant result.

Intel Compilers provide the -fp-model (Linux\* and Mac OS\* X) or /fp (Windows\*) option, which allows you to control the optimizations performed when you build an application. The option allows you to specify the compiler rules for:

 Value safety: Whether the compiler may perform transformations that could affect the result. For example, in the SAFE mode, the compiler won't transform x/x to 1.0. The UNSAFE mode is the default.

- Floating-point expression evaluation: How the compiler should handle the rounding of intermediate expressions.
- Floating-point contractions: Whether the compiler should generate floating-point multiply-add (FMA) on processors based on the IA-64 architecture.
   When enabled, the compiler may generate FMA for combined multiply/add; when disabled, the compiler must generate separate multiply/add with intermediate rounding.
- Floating-point environment access: Whether the compiler must account for the possibility that the program might access the floating-point environment, either by changing the default floating-point control settings or by reading the floating-point status flags. This is disabled by default. You can use the -fp-model:strict (Linux and Mac OS X) /fp:strict (Windows) option to enable it.
- Precise floating-point exceptions: Whether the compiler should account for
  the possibility that floating-point operations might produce an exception. This
  is disabled by default. You can use -fp-model:strict (Linux and Mac OS
  X) or /fp:strict (Windows); or -fp-model:except (Linux and Mac OS
  X) or /fp:except (Windows) to enable it.

The following table describes the impact of different keywords of the option on compiler rules and optimizations:

Keyword	Value	Floating-	Floating-	Floating-	Precise
	Safety	Point	Point	Point	Floating-
		Expression	Contractions	Environment	Point
		Evaluation		Access	Exceptions
precise	Safe	Varies	Yes	No	No
source		Source			
strict	Safe	Varies	No	Yes	Yes
fast=1	Unsafe	Unknown	Yes	No	No
(default)					

Keyword	Value	Floating-	Floating-	Floating-	Precise
	Safety	Point	Point	Point	Floating-
		Expression	Contractions	Environment	Point
		Evaluation		Access	Exceptions
fast=2	Very unsafe	Unknown	Yes	No	No
except	Unaffected	Unaffected	Unaffected	Unaffected	Yes
except-	Unaffected	Unaffected	Unaffected	Unaffected	No



It is illegal to specify the except keyword in an unsafe safety mode.

Based on the objectives of an application, you can choose to use different sets of compiler options and keywords to enable or disable certain optimizations, so that

you can get the desired result.

## **Programming Objectives of Floating-point Applications**

In general, the programming objectives of the floating-point applications fall into the following categories:

- Accuracy: The application produces that results that are close to the correct result.
- Reproducibility and portability: The application produces results that are consistent across different runs, different set of build options, different compilers, different platforms, and different architectures.
- Performance: The application produces the most efficient code possible. Based on the goal of an application, you will need to balance the tradeoffs among these objectives. For example, if you are developing a 3D graphics engine, then performance can be the most important factor to consider, and reproducibility and accuracy can be your secondary concerns.

  Intel® Compiler provides appropriate compiler options, such as the -fp-model

(Linux\* and Mac OS\* X) or /fp (Windows\*) option, which allows you to tune your

applications based on specific objectives. The compiler processes the code differently when you specify different compiler options.

In most case, an application will be much more complicated. You should select appropriate compiler options by carefully considering your programming objectives and balancing the tradeoffs among these objectives.

#### **Denormal Numbers**

A normalized number is a number for which both the exponent (including offset) and the most significant bit of the mantissa are non-zero. For such numbers, all the bits of the mantissa contribute to the precision of the representation.

The smallest normalized single precision floating-point number greater than zero is about 1.1754943<sup>-38</sup>. Smaller numbers are possible, but those numbers must be represented with a zero exponent and a mantissa whose leading bit(s) are zero, which leads to a loss of precision. These numbers are called denormalized numbers; denormals (newer specifications refer to these as subnormal numbers). Denormal computations use both hardware or operating system resources to handle them, which can cost hundreds of clock cycles.

- Denormal computations take much longer to calculate on processors based on IA-32 and Intel® 64 architectures than normal computations.
- Denormals are computed in software on processors based on the IA-64 architecture, and the computation usually requires hundreds of clock cycles, which results in excessive kernel time.

There are several ways to handle denormals and increase the performance of your application:

- Scale the values into the normalized range.
- Use a higher precision data type with a larger dynamic range.
- Flush denormals to zero.

#### See Also

Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1: Basic Architecture

Intel® Itanium® Architecture Software Developer's Manual, Volume 1: Application Architecture

Institute of Electrical and Electronics Engineers, Inc\*. (IEEE) web site for information about the current floating-point standards and recommendations

## **Floating-point Environment**

The floating-point environment is a collection of registers that control the behavior of the floating-point machine instructions and indicate the current floating-point status. The floating-point environment can include rounding mode controls, exception masks, flush-to-zero (FTZ) controls, exception status flags, and other floating-point related features.

For example, on IA-32 and Intel® 64 architectures, bit 15 of the MXCSR register enables the flush-to-zero mode, which controls the masked response to an single-instruction multiple-data (SIMD) floating-point underflow condition. The floating-point environment affects most floating-point operations; therefore, correct configuration to meet your specific needs is important. For example, the exception mask bits define which exceptional conditions will be raised as exceptions by the processor. In general, the default floating-point environment is set by the operating system. You don't need to configure the floating-point environment unless the default floating-point environment does not suit your needs.

There are several methods available if you want to modify the default floatingpoint environment. For example, you can use inline assembly, compiler built-in functions, and library functions.

Changing the default floating-point environment affects runtime results only. This does not affect any calculations which are pre-computed at compile time.

#### See Also

Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1: Basic Architecture

Intel® Itanium® Architecture Software Developer's Manual, Volume 1: Application Architecture

### **Setting the FTZ and DAZ Flags**

In Intel® processors, the flush-to-zero (FTZ) and denormals-are-zero (DAZ) flags in the MXCSR register are used to control floating-point calculations. When the FTZ and DAZ flags are enabled, the Single Instructions and Multiple Data (SIMD) floating-point computation can be accelerated, thus improving the performance of the application.

You can use the <code>-ftz</code> (Linux\* and Mac OS\* X) or <code>/Qftz</code> (Windows\*) option to flush denormal results to zero when the application is in the gradual underflow mode. This option may improve performance if the denormal values are not critical to your application's behavior.

The -ftz or /Qftz option sets or resets the FTZ and the DAZ hardware flags. The following table describes how the compiler process denormal values based on the status of the FTZ and DAZ flags:

Flag	When set to ON, the compiler	When set to OFF, the compiler	Supported on
FTZ	Sets denormal results from floating-point calculations to zero.	Does not change the denormal results.	IA-64 and Intel® 64 architectures
DAZ	Treats denormal values used as input to floating-point instructions as zero.	Does not change the denormal instruction inputs.	Intel® 64 architecture

 FTZ and DAZ are not supported on all IA-32 architectures. On systems based on the IA-64 architecture, FTZ always works, while on systems based on the IA-32 and Intel® 64 architectures, it only applies to SSE instructions. Hence if you application happened to generate denormals using x87 instructions, FTZ does not apply.  DAZ and FTZ flags are not compatible with IEEE Standard 754, so you should only consider enabling them when strict compliance to the IEEE standard is not required and application performance has higher priority than application accuracy.

Options -ftz and /Qftz are performance options. Setting these options does not guarantee that all denormals in a program are flushed to zero. They only cause denormals generated at run time to be flushed to zero.

When -ftz or /Qftz is used in combination with an SSE-enabling option on systems based on the IA-32 architecture (for example, -xW or /QxW), the compiler will insert code in the main routine to set FTZ and DAZ. When -ftz or /Qftz is used without such an option, the compiler will insert code to conditionally set FTZ/DAZ based on a run-time processor check. -no-ftz (Linux and Mac OS X) or /Qftz- (Windows) will prevent the compiler from inserting any code that might set FTZ or DAZ.

The -ftz or /Qftz option only has an effect when the main program is being compiled. It sets the FTZ/DAZ mode for the process. The initial thread and any threads subsequently created by that process will operate in the FTZ/DAZ mode. On systems based on the IA-64 architecture, optimization option O3 sets -ftz and /Qftz; optimization option O2 sets -no-ftz (Linux) and /Qftz- (Windows). On systems based on the IA-32 and Intel® 64 architectures, every optimization option O level, except O0, sets -ftz and /Qftz-. If this option produces undesirable results of the numerical behavior of your

If this option produces undesirable results of the numerical behavior of your program, you can turn the FTZ/DAZ mode off by using -no-ftz or /Qftz- in the command line while still benefiting from the O3 optimizations.

For some non-Intel processors, the flags can be set manually by calling the following Intel Fortran intrinsic:

### Example

RESULT = FOR\_SET\_FPE (FOR\_M\_ABRUPT\_UND)

#### See Also

Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1: Basic Architecture

### **Tuning Performance**

### **Overview: Tuning Performance**

This section describes several programming guidelines that can help you improve the performance of a floating-point applications:

- Avoid exact floating-point comparisons
- Avoid exceeding representable ranges during computation; handling these cases can have a performance impact.
- Use REAL variables in single precision format unless the extra precision obtained through DOUBLE or REAL\*8 is required because a larger precision formation will also increase memory size and bandwidth requirements.
- Reduce the impact of denormal exceptions for all supported architectures.
- Avoid mixed data type arithmetic expressions.

## **Avoiding Exact Floating-point Comparison**

It is unsafe for applications to rely on exact floating-point comparisons. Slight variations in rounding can change the outcome of such comparisons, leading to non-convergence or other unexpected behavior.

Tests for equality of floating-point quantities should be made within some tolerance related to the expected precision of the calculation, for example, by using the Fortran intrinsic function EPSILON. The following examples demonstrate the concept:

# Example

$$if (foo() == 2.0)$$

Where foo() may be as close to 2.0 as can be imagined without actually exactly matching 2.0. You can improve the behavior of such codes by using inexact floating-point comparisons or fuzzy comparisons to test a value to within a certain tolerance, as shown below:

# Example

```
epsilon = 1E-8;
if (abs(foo() - 2.0) <= epsilon)
```

# Handling Floating-point Array Operations in a Loop Body

Following the guidelines below will help autovectorization of the loop.

- Statements within the loop body may contain float or double operations (typically on arrays). The following arithmetic operations are supported: addition, subtraction, multiplication, division, negation, square root, MAX, MIN, and mathematical functions such as SIN and COS.
- Writing to a float scalar/array and a double scalar/array within the same loop decreases the chance of autovectorization due to the differences in the vector length (that is, the number of elements in the vector register) between float and double types. If autovectorization fails, try to avoid using mixed data types.

# **Reducing the Impact of Denormal Exceptions**

Denormalized floating-point values are those that are too small to be represented in the normal manner; for example, the mantissa cannot be left-justified.

Denormal values require hardware or operating system interventions to handle the computation, so floating-point computations that result in denormal values may have an adverse impact on performance.

There are several ways to handle denormals to increase the performance of your application:

- Scale the values into the normalized range
- Use a higher precision data type with a larger dynamic range
- Flush denormals to zero

For example, you can translate them to normalized numbers by multiplying them using a large scalar number, doing the remaining computations in the normal space, then scaling back down to the denormal range. Consider using this method when the small denormal values benefit the program design.

If you change the declaration of a variable you might also need to change the libraries you call to use the variable Another strategy that might result in

increased performance is to increase the amount of precision of intermediate values using the <code>-fp-model [double|extended]</code> option; however, if you increased precision as the solution to slow performance caused by denormal numbers you must verify the resulting changes actually increase performance. Finally, In many cases denormal numbers be treated safely as zero without adverse effects on program results. Depending on the target architecture, use flush-to-zero (FTZ) options.

#### IA-32 and Intel® 64 Architectures

These architectures take advantage of the FTZ and DAZ (denormals-are-zero) capabilities of Streaming SIMD Extensions (SSE), Streaming SIMD Extensions 2 (SSE2), and Streaming SIMD Extensions 3 (SSE3), and Supplemental Streaming SIMD Extensions 3 (SSSE3) instructions.

By default, the compiler for the IA-32 architecture generates code that will run on machines that do not support SSE instructions. The compiler implements floating-point calculations using the x87 floating-point unit, which does not benefit from the FTZ and DAZ settings. You can use the -x (Linux\* and Mac OS\* X) or /Qx (Windows\*) option to enable the compiler to implement floating-point calculations using the SSE and SSE2 instructions. The compiler for the Intel® 64 architecture generates SSE2 instructions by default.

The FTZ and DAZ modes are enabled by default when you compile the source file containing PROGRAM using the Intel Compiler. The compiler generates a call to a library routine that performs a runtime processor check. The FTZ and DAZ modes are set provided that the modes are available for the machine on which the program is running.

#### **IA-64 Architecture**

Enable the FTZ mode by using the -ftz (Linux and Mac OS X) or /Qftz (Windows) option on the source file containing PROGRAM. The -O3 (Linux and Mac OS X) or /O3 (Windows) option automatically enables -ftz or /Qftz.



After using flush-to-zero, ensure that your program still gives correct results when treating denormalized values as zero.

#### See Also

# Setting the FTZ and DAZ Flags

Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1: Basic Architecture

# **Avoiding Mixed Data Type Arithmetic Expressions**

Avoid mixing integer and floating-point (REAL) data in the same computation. Expressing all numbers in a floating-point arithmetic expression (assignment statement) as floating-point values eliminates the need to convert data between fixed and floating-point formats. Expressing all numbers in an integer arithmetic expression as integer values also achieves this. This improves run-time performance.

For example, assuming that I and J are both INTEGER variables, expressing a constant number (2.) as an integer value (2) eliminates the need to convert the data. The following examples demonstrate inefficient and efficient code.

# **Example: Inefficient Code**

```
INTEGER I, JI = J / 2.
```

# **Example: Inefficient Code**

```
INTEGER I, J
I = J / 2
```

You can use different sizes of the same general data type in an expression with minimal or no effect on run-time performance. For example, using REAL, DOUBLE PRECISION, and COMPLEX floating-point numbers in the same

floating-point arithmetic expression has minimal or no effect on run-time performance. However, this practice of mixing different sizes of the same general data type in an expression can lead to unexpected results due to operations being performed in a lower precision than desired.

# **Using Efficient Data Types**

In cases where more than one data type can be used for a variable, consider selecting the data types based on the following hierarchy, listed from most to least efficient:

- Integer
- Single-precision real, expressed explicitly as REAL, REAL (KIND=4), or REAL\*4
- Double-precision real, expressed explicitly as DOUBLE PRECISION, REAL (KIND=8), or REAL\*8
- Extended-precision real, expressed explicitly as REAL (KIND=16) or REAL\*16

However, keep in mind that in an arithmetic expression, you should avoid mixing integer and floating-point data.

# **Handling Floating-point Exceptions**

# **Overview: Controlling Floating-point Exceptions**

When developing applications, you may need to control the exceptions that can occur during the run-time processing of floating-point numbers. These exceptions can be categorized into specific types: overflow, divide-by-zero, underflow, and invalid operations.

#### Overflow

Overflow is signaled whenever the destination format's largest finite number is exceeded in magnitude by what would have been the rounded floating-point result. The result computed is rounding mode specific:

Round-to-nearest (default): +/- Infinity in specified precision

- Round-to-zero: +/- Maximum Number in specified precision
- Round-to-+Infinity: +Infinity or -(Maximum Positive Number) in specified precision
- Round-to--Infinity: (Maximum Positive Number) or -Infinity in specified precision

For example, in round-to-nearest mode 1E30 \* 1E30 overflows the single-precision floating-point range and results in a +Infinity; -1E30 \* 1E30 results in a -Infinity.

#### Divide-by-zero

Divide-by-zero is signaled when the divisor is zero and the dividend is a finite nonzero number. The computed result is a correctly signed Infinity. For example, 2.0E0/+0.0 produces a divide-by-zero exception and results in a +Infinity; -2.0E0/+0.0 produces a divide-by-zero exception and results in a -Infinity.

#### Underflow

Underflow occurs when a computed result (of an add, subtract, multiply, divide, or math function call) falls beyond the minimum range in magnitude of normalized numbers of the floating-point data type. Each floating-point type (32-, 64-, and 128-bit) has a denormalized range where very small numbers can be represented with some loss of precision. This is called gradual underflow. For example, the lower bound for normalized single-precision floating-point is approximately 1E-38, while the lower bound for denormalized single-precision floating-point is approximately 1E-45. Results falling below the lower bound of the denormalized range simply become zero. 1E-30 / 1E10 underflows the normalized range but not the denormalized range so the result is the denormal value 1E-40. 1E-30 / 1E30 underflows the entire range and the result is zero.

#### Invalid operation

Invalid occurs when operands to the basic floating-point operations or math function inputs produce an undefined (QNaN) result. Some examples include:

- SNaN operand in any floating-point operation or math function call
- Division of zeroes: (+/-0.0)/(+/-0.0)
- Sum of Infinities having different signs: Infinity + (-Infinity)
- Difference of Infinities having the same sign: (+/-Infinity) (+/-Infinity)
- Product of signed Infinities with zero: (+/-Inf) \* 0
- Math Function Domain Errors: log(negative), sqrt(negative), asin(|x}>1)

With the Intel® Compiler, you can use the -fpe (Linux\* and Mac OS\* X) or /fpe (Windows\*) option to control floating-point exceptions.

# **Handling Floating-point Exceptions**

If a floating-point exception is disabled (its bit is set to 1 with SETCONTROLFPQQ (IA-32 architecture only), it will not generate an interrupt signal if it occurs. The floating-point process may return an appropriate special value (for example, NaN or signed infinity) or may return an acceptable value (for example, in the case of a denormal operand), and the program will continue. If a floating-point exception is enabled (its bit is set to 0), it will generate an interrupt signal (software interrupt) if it occurs.

The following table lists the floating-point exception signals:

Parameter Name	Value in Hex	Description
FPE\$INVALID	#81	Invalid result
FPE\$DENORMAL	#82	Denormal operand
FPE\$ZERODIVIDE	#83	Divide by zero
FPE\$OVERFLOW	#84	Overflow
FPE\$UNDERFLOW	#85	Underflow
FPE\$INEXACT	#86	Inexact precision

If a floating-point exception interrupt occurs and you do not have an exception handling routine, the run-time system will respond to the interrupt according to the behavior selected by the compiler option /fpe. Remember, interrupts only occur if an exception is enabled (set to 0).

If you do not want the default system exception handling, you need to write your own interrupt handling routine:

- Write a function that performs whatever special behavior you require on the interrupt.
- Register that function as the procedure to be called on that interrupt with SIGNALQQ.

Note that your interrupt handling routine must use the cDEC\$ ATTRIBUTES option C.

The drawback of writing your own routine is that your exception-handling routine cannot return to the process that caused the exception. This is because when your exception-handling routine is called, the floating-point processor is in an error condition, and if your routine returns, the processor is in the same state, which will cause a system termination. Your exception-handling routine can therefore either branch to another separate program unit or exit (after saving your program state and printing an appropriate message). You cannot return to a different statement in the program unit that caused the exception-handling routine, because a global GOTO does not exist, and you cannot reset the status word in the floating-point processor.

If you need to know when exceptions occur and also must continue if they do, you must disable exceptions so they do not cause an interrupt, then poll the floating-point status word at intervals with GETSTATUSFPQQ (IA-32 architecture only) to see if any exceptions occurred. To clear the status word flags, call the CLEARSTATUSFPQQ (IA-32 architecture only) routine.

Polling the floating-point status word at intervals creates processing overhead for your program. In general, you will want to allow the program to terminate if there is an exception. An example of an exception-handling routine follows. The comments at the beginning of the SIGTEST.F90 file describe how to compile this example.

```
! SIGTEST.F90
!Establish the name of the exception handler as the
```

```
! function to be invoked if an exception happens.
 ! The exception handler hand_fpe is attached below.
  USE IFPORT
     INTERFACE
       FUNCTION hand_fpe (sigid, except)
          !DEC$ ATTRIBUTES C :: hand fpe
          INTEGER(4) hand fpe
          INTEGER(2) sigid, except
       END FUNCTION
     END INTERFACE
INTEGER(4) iret
REAL(4) r1, r2
r1 = 0.0
iret = SIGNALQQ(SIG$FPE, hand_fpe)
WRITE(*,*) 'Set exception handler. Return = ', iret
! Cause divide-by-zero exception
r1 = 0.0
r2 = 3/r1
END
! Exception handler routine hand_fpe
  FUNCTION hand_fpe (signum, excnum)
    !DEC$ ATTRIBUTES C :: hand_fpe
    USE IFPORT
    INTEGER(2) signum, excnum
    WRITE(*,*) 'In signal handler for SIG$FPE'
    WRITE(*,*) 'signum = ', signum
    WRITE(*,*) 'exception = ', excnum
    SELECT CASE (excnum)
      CASE( FPE$INVALID )
        STOP ' Floating point exception: Invalid number'
      CASE ( FPE$DENORMAL )
        STOP ' Floating point exception: Denormalized number'
      CASE( FPE$ZERODIVIDE )
        STOP ' Floating point exception: Zero divide'
      CASE ( FPE$OVERFLOW )
        STOP ' Floating point exception: Overflow'
      CASE ( FPE$UNDERFLOW )
        STOP ' Floating point exception: Underflow'
      CASE( FPE$INEXACT )
        STOP ' Floating point exception: Inexact precision'
      CASE DEFAULT
        STOP ' Floating point exception: Non-IEEE type'
    END SELECT
    hand fpe = 1
  END
```

# **Checking the Floating-point Stack State**

On systems based on the IA-32 architectures, when an application calls a function that returns a floating-point value, the returned floating-point value is supposed to be on the top of the floating-point stack. If the return value is not

used, the compiler must pop the value off of the floating-point stack in order to keep the floating-point stack in the correct state.

On systems based on Intel(R) 64 architectures, floating-point values are usually returned in the xmm0 register. The floating-point stack is used only when the return value is a long doublean internal 80-bit floating-point data type on Linux\* and Mac OS\* X systems.

If the application calls a function without defining or incorrectly defining the function's prototype, the compiler cannot determine if the function must return a floating-point value. Consequently, the return value is not popped off the floating-point stack if it is not used. This can cause the floating-point stack to overflow.

The overflow of the stack results in two undesirable situations:

- A NaN value gets involved in the floating-point calculations
- The program results become unpredictable; the point where the program starts making errors can be arbitrarily far away from the point of the actual error.

For systems based on the IA-32 and Intel® 64 architectures, the -fp-stack-check (Linux\* and Mac OS\* X) or /Qfp-stack-check (Windows\*) option checks whether a program makes a correct call to a function that should return a floating-point value. If an incorrect call is detected, the option places a code that marks the incorrect call in the program. The -fp-stack-check (Linux\* and Mac OS\* X) or /Qfp-stack-check (Windows\*) option marks the incorrect call and makes it easy to find the error.

# Note

The -fp-stack-check (Linux\* and Mac OS\* X) and the /Qfp-stack-check (Windows\*) option causes significant code generation after every function/subroutine call to ensure that the floating-point stack is maintained in the correct state. Therefore, using this option slows down the program being compiled. Use the option only as a debugging aid to find floating point stack underflow/overflow problems, which can be otherwise hard to find.

#### See Also

<u>-fp-stack-check, /Qfp-stack-check option</u> Tells the compiler to generate extra code after every function call to ensure that the floating-point stack is in the expected state.

# File fordef.for and Its Usage

The parameter file <code>fordef.for</code> contains symbols and INTEGER\*4 values corresponding to the classes of floating-point representations. Some of these classes are exceptional ones such as bit patterns that represent positive denormalized numbers.

With this file of symbols and with the FP\_CLASS intrinsic function, you have the flexibility of identifying exceptional numbers so that, for example, you can replace positive and negative denormalized numbers with true zero.

The following is a simple example of identifying floating-point bit representations:

In this example, the symbol for\_k\_fp\_pos\_norm in the file fordef.for plus the REAL\*4 value 57.0 to the FP\_CLASS intrinsic function results in the execution of the first print statement.

The table below explains the symbols in the file *fordef.for* and their corresponding floating-point representations.

### Symbols in fordef.for

Symbol Name	Class of Floating-Point Bit Representation
FOR_K_FP_SNAN	Signaling NaN
FOR_K_FP_QNAN	Quiet NaN
FOR_K_FP_POS_INF	Positive infinity

Symbol Name	Class of Floating-Point Bit Representation
FOR_K_FP_NEG_INF	Negative infinity
FOR_K_FP_POS_NORM	Positive normalized finite number
FOR_K_FP_NEG_NORM	Negative normalized finite number
FOR_K_FP_POS_DENOR	Positive denormalized number
FOR_K_FP_NEG_DENOR	M Negative denormalized number
FOR_K_FP_POS_ZERO	Positive zero
FOR_K_FP_NEG_ZERO	Negative zero

Another example of using file <code>fordef.for</code> and intrinsic function FP\_CLASS follows. The goals of this program are to quickly read any 32-bit pattern into a REAL\*4 number from an unformatted file with no exception reporting and to replace denormalized numbers with true zero:

You can compile this program with any value of -fpen (Linux\* and Mac OS\* X) or /fpe:n (Windows\*). Intrinsic function FP\_CLASS helps to find and replace denormalized numbers with zeroes before the program can attempt to perform calculations on the denormalized numbers.

On the other hand, if this program did not replace denormalized numbers read from unit 1 with zeroes and the program was compiled with -fpe0 or /fpe:0, then the first attempted calculation on a denormalized number would result in a floating-point exception. If you compile with /fpe:0 flush-to-zero is enabled. If

the resulting calculation creates a divide-by-zero, overflow, or invalid operation, then the application should abort with a floating-point exception. Otherwise, a program using the data will run to completion, perhaps faster and with different answers.

File <code>fordef.for</code> and intrinsic function <code>FP\_CLASS</code> can work together to identify NaNs. A variation of the previous example would contain the symbols <code>for\_k\_fp\_snan</code> and <code>for\_k\_fp\_qnan</code> in the IF statement. A faster way to do this is based on the intrinsic ISNAN function. One modification of the previous example, using ISNAN, follows:

```
! The ISNAN function does not need file fordef.for
real*4 a(100)
! open an unformatted file as unit 1
! ...
read (1) a
do i = 1, 100
   if ( isnan (a(i)) ) then
      print *, 'Element ', i, ' contains a NaN'
   end if
end do
close (1)
end
```

You can compile this program with any value of -fpen or /fpe:n.

# **Setting and Retrieving Floating-point Status and Control Words (IA-32)**

# Overview: Setting and Retrieving Floating-point Status and Control Words

The FPU (floating-point unit) on systems based on the IA-32 architecture contains eight floating-point registers the system uses for numeric calculations, status and control words, and error pointers. You normally need to consider only the status and control words, and then only when customizing your floating-point environment.

The FPU status and control words correspond to 16-bit registers whose bits hold the value of a state of the FPU or control its operation. Intel Fortran defines a set of symbolic constants to set and reset the proper bits in the status and control words.

Note

The symbolic constants and the library routines used to read and write the control and status registers only affect the x87 control and status registers. They do not affect the MXCSR register (the control and status register for the Intel(R) SSE and Intel(R) SSE2 instructions).

They do not affect the MXCSR (the control and status register for the Intel(R) SSE and Intel(R) SSE2 instructions). For example:

```
USE IFPORT
INTEGER(2) status, control, controlo, mask_all_traps
CALL GETCONTROLFPQQ(control)
WRITE (*, 9000) 'Control word: ', control
! Save old control word
controlo = control
! Clear the rounding control flags
control = IAND(control,NOT(FPCW$MCW_RC))
! Set new control to round up
control = IOR(control,FPCW$UP)
CALL SETCONTROLFPQQ(control)
CALL GETCONTROLFPQQ(control)
WRITE (*, 9000) 'Control word: ', control
! Demonstrate setting and clearing exception mask flags
mask_all_traps = FPCW$INVALID + FPCW$DENORMAL + &
   FPCW$ZERODIVIDE + FPCW$OVERFLOW + &
FPCW$UNDERFLOW + FPCW$INEXACT
     Clear the exception mask flags
control = IAND(control,NOT(FPCW$MCW_EM))
     Set new exception mask to disallow overflow
     (i.e., enable overflow traps)
! but allow (i.e., mask) all other exception conditions.
control = IOR(control,IEOR(mask_all_traps,FPCW$OVERFLOW))
CALL SETCONTROLFPQQ(control)
CALL GETCONTROLFPQQ(control)
WRITE (*, 9000) 'Control word: ', control
9000 FORMAT (1X, A, Z4)
END
```

The status and control symbolic constants (such as FPCW\$OVERFLOW and FPCW\$CHOP in the preceding example) are defined as INTEGER(2) parameters in the module IFORT.F90 in the . . . \ INCLUDE folder. The status and control words are logical combinations (such as with .AND.) of different parameters for different FPU options.

The name of a symbolic constant takes the general form *name*\$option. The prefix *name* is one of the following:

**Prefixes for Parameter Flags** 

# Intel® Fortran Compiler User and Reference Guides

name	Meaning
FPSW	Floating-point status word
FPCW	Floating-point control word
SIG	Signal
FPE	Floating-point exception
MTH	Math function

The suffix *option* is one of the options available for that *name*. The parameter *name*\$*option* corresponds either to a status or control option (for example, FPSW\$ZERODIVIDE, a status word parameter that shows whether a zero-divide exception has occurred or not) or *name*\$*option* corresponds to a mask, which sets all symbolic constants to 1 for all the options of *name*. You can use the masks in logical functions (such as IAND, IOR, and NOT) to set or to clear all options for the specified *name*. The following sections define the *option*s and illustrate their use with examples.

You can control the floating-point processor options (on systems based on the IA-32 architecture) and find out its status with the run-time library routines GETSTATUSFPQQ (IA-32 architecture only), GETCONTROLFPQQ (IA-32 architecture only), and SETCONTROLFPQQ (IA-32 architecture only). Examples of using these routines also appear in the following sections.

#### See Also

<u>Understanding Floating-Point Status Word (IA-32 architecture only)</u>
<u>Understanding Floating-Point Control Word (IA-32 architecture only)</u>

# **Understanding Floating-point Status Word**

On systems based on the IA-32 architecture, the FPU status word includes bits that show the floating-point exception state of the processor. The status word parameters describe six exceptions: invalid result, denormalized operand, zero divide, overflow, underflow and inexact precision. These are described in the section, Loss of Precision Errors. When one of the bits is set to 1, it means a past

floating-point operation produced that exception type. (Intel Fortran initially clears all status bits. It does not reset the status bits before performing additional floating-point operations after an exception occurs. The status bits accumulate.) The following table shows the floating-point exception status parameters:

Parameter Name	Value in Hex	Description
FPSW\$MSW_EM	#003F	Status Mask (set all bits to 1)
FPSW\$INVALID	#0001	An invalid result occurred
FPSW\$DENORMAL	#0002	A denormal operand occurred
FPSW\$ZERODIVIDE	#0004	A divide by zero occurred
FPSW\$OVERFLOW	#0008	An overflow occurred
FPSW\$UNDERFLOW	#0010	>An underflow occurred
FPSW\$INEXACT	#0020	Inexact precision occurred

You can find out which exceptions have occurred by retrieving the status word and comparing it to the exception parameters. For example:

#### USE IFPORT

To clear the status word flags, call the CLEARSTATUSFPQQ (IA-32 architecture only) routine.



The GETSTAUSFPQQ and CLEARSTATUSFPQQ routines only affect the x87 status register. They do not affect the MXCSR register (the control and status register for the Intel(R) SSE and Intel(R) SSE2 instructions).

## **Floating-point Control Word Overview**

On systems based on the IA-32 architecture, the FPU control word includes bits that control the FPU's precision, rounding mode, and whether exceptions

generate signals if they occur. You can read the control word value with GETCONTROLFPQQ (IA-32 architecture only) to find out the current control settings, and you can change the control word with SETCONTROLFPQQ (IA-32 architecture only).

# Note

The GETCONTROLFPQQ and SETCONTROLFPQQ routines only affect the x87 status register. They do not affect the MXCSR register (the control and status register for the SSE and SSE2 instructions).

Each bit in the floating-point control word corresponds to a mode of the floating-point math processor. The *IFORT.F90* module file in the . . . \ *INCLUDE* folder contains the INTEGER(2) parameters defined for the control word, as shown in the following table:

Parameter Name	Value in Hex	Description		
FPCW\$MCW_PC	#0300	Precision control mask		
FPCW\$64	#0300	64-bit precision		
FPCW\$53	#0200	53-bit precision		
FPCW\$24	#0000	24-bit precision		
FPCW\$MCW_RC	#0000	Rounding control mask		
FPCW\$CHOP	#0000	Truncate		
FPCW\$UP	#0800	Round up		
FPCW\$DOWN	#0400	Round down		
FPCW\$NEAR	#0000	Round to nearest		
FPCW\$MCW_EM	#003F	Exception mask		
FPCW\$INVALID	#0001	Allow invalid numbers		
>FPCW\$DENORMAL	#0002	Allow denormals (very small numbers)		
FPCW\$ZERODIVIDE	#0004	Allow divide by zero		

Parameter Name	Value in Hex	Description
FPCW\$OVERFLOW	#0008	Allow overflow
FPCW\$UNDERFLOW	#0010	Allow underflow
FPCW\$INEXACT	#0020	Allow inexact precision

## The control word defaults are:

- 53-bit precision
- Round to nearest (rounding mode)
- The denormal, underflow, overflow, divide-by-zero, invalid, and inexact precision exceptions are disabled (do not generate an exception). To change exception handling, you can use the -fpe (Linux\* and Mac OS\* X) or the /fpe (Windows\*) compiler option or the FOR\_SET\_FPE routine.

# Using Exception, Precision, and Rounding Parameters

This topic describes the exception, precision, and rounding parameters that you can use for the control word.

#### **Exception Parameters**

An exception is disabled if its bit is set to 1 and enabled if its bit is cleared to 0. If an exception is disabled (exceptions can be disabled by setting the flags to 1 with SETCONTROLFPQQ [IA-32 architecture only]), it will not generate an interrupt signal if it occurs. The floating-point process will return an appropriate special value (for example, NaN or signed infinity), but the program continues. You can find out which exceptions (if any) occurred by calling GETSTATUSFPQQ (IA-32 architecture only).

If errors on floating-point exceptions are enabled (by clearing the flags to 0 with SETCONTROLFPQQ [IA-32 architecture only]), the operating system generates an interrupt when the exception occurs. By default these interrupts cause runtime errors, but you can capture the interrupts with SIGNALQQ and branch to your own error-handling routines.

You should remember not to clear all existing settings when changing one. The values you want to change should be combined with the existing control word in an inclusive-OR operation (IOR) if you do not want to reset all options. For example:

```
USE IFPORT
INTEGER(2) control, newcontrol
CALL GETCONTROLFPQQ(control)
newcontrol = IOR(control,FPCW$INVALID)
! Invalid exception set (disabled).
CALL SETCONTROLFPQQ(newcontrol)
```

Note

The GETCONTROLFPQQ, SETCONTROLFPQQ, and GETSTATUSFPQQ routines only affect the x87 status register. They do not affect the MXCSR register (the control and status register for the Intel(R) SSE and Intel(R) SSE2 instructions).

#### **Precision Parameters**

On systems based on the IA-32 architecture, the precision bits control the precision to which the FPU rounds floating-point numbers. For example:

```
USE IFPORT
INTEGER(2) control, holdcontrol, newcontrol
CALL GETCONTROLFPQQ(control)
! Clear any existing precision flags.
holdcontrol = IAND(control, NOT(FPCW$MCW_PC))
newcontrol = IOR(holdcontrol, FPCW$64)
! Set precision to 64 bits.
CALL SETCONTROLFPQQ(newcontrol)
```

The precision options are mutually exclusive. If you set more than one, you may get an invalid mode or a mode other than the one you want. Therefore, you should clear the precision bits before setting a new precision mode.



The GETCONTROLFPQQ and SETCONTROLFPQQ routines only affect the x87 status register. They do not affect the MXCSR register (the control and status register for the Intel(R) SSE and Intel(R) SSE2 instructions).

#### **Rounding Parameters**

On systems based on the IA-32 architecture, the rounding flags control the method of rounding that the FPU uses. For example:

```
USE IFPORT
INTEGER(2) status, control, controlo, mask_all_traps
CALL GETCONTROLFPQQ(control)
WRITE (*, 9000) 'Control word: ', control
>
! Save old control word
controlo = control
! Clear the rounding control flags
control = IAND(control,NOT(FPCW$MCW_RC))
! Set new control to round up
control = IOR(control,FPCW$UP)
CALL SETCONTROLFPQQ(control)
CALL GETCONTROLFPQQ(control)
WRITE (*, 9000) 'Control word: ', control
```

The rounding options are mutually exclusive. If you set more than one, you may get an invalid mode or a mode other than the one you want. Therefore, you should clear the rounding bits before setting a new rounding mode.



The GETCONTROLFPQQ and SETCONTROLFPQQ routines only affect the x87 status register. They do not affect the MXCSR register (the control and status register for the Intel(R) SSE and Intel(R) SSE2 instructions).

# Handling Floating-point Exceptions with the -fpe or /fpe Compiler Option Using the -fpe or /fpe Compiler Option

The -fpen (Linux\* and Mac OS\* X) or /fpe:n (Windows\*) option allows some control over the results of floating-point exceptions.

-fpe0 or /fpe:0 restricts floating-point exceptions by enabling the overflow, the divide-by-zero, and the invalid floating-point exceptions. The program will print an error message and abort if any of these exceptions occurs. If a floating underflow occurs, the result is set to zero and execution continues. This is called flush-to-zero. This option sets -IPF\_fp\_speculationstrict (Linux and Mac OS X) or /QIPF\_fp\_speculationstrict (Windows) if no specific -IPF\_fp\_speculation or /QIPF\_fp\_speculation option is specified. The -fpe0 or /fpe:0 option sets -ftz (Linux and Mac OS X) /Qftz(Windows). To

get more detailed location information about where the exception occurred, use – traceback (Linux and Mac OS X) or /traceback (Windows).

# Note

On systems based on the IA-32 and Intel® 64 architectures, explicitly setting -fpe0 or /fpe:0 can degrade performance since the generated code stream must be synchronized after each floating-point instruction to allow for abrupt underflow fix-up.

-fpe1 or /fpe:1 restricts only floating-point underflow. Floating-point overflow, floating-point divide-by-zero, and floating-point invalid produce exceptional values (NaN and signed Infinities) and execution continues. If a floating-point underflow occurs, the result is set to zero and execution continues. The /fpe:1 option sets -ftz or /Qftz.

# Note

On systems based on the IA-32 and Intel® 64 architectures , explicitly setting -fpe1 or /fpe:1 can degrade performance since the generated code stream must be synchronized after each floating-point instruction to allow for abrupt underflow fix-up.

-fpe3 or /fpe:3 is the default on all processors, which allows full floating-point exception behavior. Floating-point overflow, floating-point divide-by-zero, and floating-point invalid produce exceptional values (NaN and signed Infinities) and execution continues. Floating underflow is gradual: denormalized values are produced until the result becomes 0.

The -fpe or /fpe option enables exceptions in the Fortran main program only. The floating-point exception behavior set by the Fortran main program remains in effect throughout the execution of the entire program unless changed by the programmer. If the main program is not Fortran, the user can use the Fortran intrinsic FOR\_SET\_FPE to set the floating-point exception behavior. When compiling different routines in a program separately, you should use the same value of n in -fpen or /fpe:n.

An example follows:

IMPLICIT NONE

```
real*4 res_uflow, res_oflow
      real*4 res_dbyz, res_inv
      real*4 small, big, zero, scale
      small = 1.0e-30
      big = 1.0e30
      zero = 0.0
      scale = 1.0e-10
!
    IEEE underflow condition (Underflow Raised)
      res_uflow = small * scale
      write(6,100)"Underflow: ",small, " *", scale, " = ", res_uflow
!
    IEEE overflow condition (Overflow Raised)
     res_oflow = big * big
     write(6,100)"Overflow:", big, " *", big, " = ", res_oflow
!
     IEEE divide-by-zero condition (Divide by Zero Raised)
     res_dbyz = -big / zero
      write(6,100)"Div-by-zero:", -big, " /", zero, " = ", res_dbyz
     IEEE invalid condition (Invalid Raised)
      res_inv = zero / zero
      write(6,100)"Invalid:", zero, " /", zero, " = ", res_inv
100 format(A14,E8.1,A2,E8.1,A2,E10.1)
```

## Consider the following command line:

```
ifort fpe.f90 -fpe0 -fp-model strict -g (Linux and Mac OS X) ifort fpe.f90 /fpe:0 /fp:strict /traceback (Windows)
```

# Output similar to the following should result:

## Windows:

```
Underflow: 0.1E-29 * 0.1E-09 = 0.0E+00

forrtl: error (72): floating overflow

Image PC Routine Line Source

fpe.exe 0040115B Unknown Unknown Unknown

fpe.exe 0044DFC0 Unknown Unknown Unknown

fpe.exe 00433277 Unknown Unknown Unknown

kernel32.dll 7C816D4F Unknown Unknown
```

#### Linux and Mac OS X:

```
./a.out
   Underflow: 0.1E-29* 0.1E-09 = 0.0E+00
forrtl: error (72): floating overflow
Image   PC   Routine   Line   Source
a.out   0804A063  Unknown   Unknown  Unknown
a.out   08049E78  Unknown   Unknown  Unknown
Unknown   B746B748  Unknown   Unknown  Unknown
a.out   08049D31  Unknown  Unknown  Unknown
Aborted
```

#### The following command line uses /fpe1:

```
ifort fpe.f90 -fpe1 -g (Linux and Mac OS X)
ifort fpe.f90 /fpe:1 /traceback (Windows)
```

### The following output is produced:

```
Underflow: 0.1E-29 * 0.1E-09 = 0.0E+00
```

# Intel® Fortran Compiler User and Reference Guides

```
Overflow: 0.1E+31 * 0.1E+31 = Infinity
Div-by-zero: -0.1E+31 / 0.0E+00 = -Infinity
Invalid: 0.0E+00 / 0.0E+00 = NaN
```

The following command line uses /fpe3:

```
ifort fpe.f90 -fpe3 -g (Linux and Mac OS X)
ifort fpe.f90 /fpe:3 /traceback (Windows)
```

# The following output is produced:

```
Underflow: 0.1E-29 * 0.1E-09 = 0.1E-39
   Overflow: 0.1E+31 * 0.1E+31 = Infinity
Div-by-zero: -0.1E+31 / 0.0E+00 = -Infinity
   Invalid: 0.0E+00 / 0.0E+00 = NaN
```

#### **Understanding the Impact of Application Types**

The full consequences of the /fpe option depend on the type of application you are building. You only get the full support for the chosen option setting in a Fortran Console or QuickWin/Standard Graphics application, assuming you do not override the default run-time exception handler. The work to achieve the full behavior is done partly by each of the default run-time handler, the Fortran compiler, the math library, the underlying hardware, and the operating system.

Floating-point Exceptions in Fortran Console, Fortran QuickWin, and Fortran Standard Graphics Applications

When you build a console application, the compiler generates a few calls at the beginning of your Fortran main program to Fortran run-time routines that initialize the environment, either with default options or in accordance with your selected compile-time options.

For floating-point exception handling, /fpe:3 is the default. The run-time system initializes the hardware to mask all exceptions. With /fpe:3:

- The Intel Fortran run-time system does this initialization automatically (no call from the compiled code).
- The IA-32 architecture hardware automatically generates the default IEEE result for exceptional conditions.
- Because traps are masked with /fpe:3, there are no traps and you may see exceptional values like Nan's, Infinities, and denormalized numbers in your computations.

Users can poll the <u>floating-point status word</u> with GETSTATUSFPQQ to see if an exception has occurred and can clear the status register with CLEARSTATUSFPQQ.

If you specify <code>/fpe:0</code>, the compiler generates a call to an Intel Fortran run-time routine <code>FOR\_SET\_FPE</code>, with an argument that unmasks all floating-point traps in the <code>floating-point control word</code>. In this case, the hardware does not supply the default <code>IEEE</code> result. It traps to the operating system, which then looks for a condition handler.

In a Fortran console or Fortran QuickWin application, the Intel Fortran run-time system provides a default condition handler unless you establish your own. For all exceptions except underflow, the run-time system just prints out an error message and aborts the application. For underflow, the run-time system replaces the result with zero. This treatment of underflow with fpe:0 is called *abrupt underflow* to 0 (zero), as opposed to *gradual underflow* to 0 provided with fpe:3.

Fixing up underflow results to zero can significantly degrade the performance of your application based on IA-32 architecture. If you are experiencing a large number of underflows, consider changing your code to avoid underflows or consider masking underflow traps and allowing the hardware to operate on denormalized numbers. The IA-32 architecture based hardware is designed to operate correctly in the denormalized range and doing so is much faster than trapping to fix up a result to zero.

Another important point to understand about selecting the <code>/fpe</code> option is that the generated code must support the trapping mode. When an IA-32 architecture based floating-point instruction generates an exceptional result, you do not necessarily get a trap. The instruction must be followed by an <code>fwait</code> instruction or another floating-point operate instruction to cause the trap to occur. The Intel Fortran compiler generates machine code to support these requirements in accordance with your selected <code>/fpe</code> option setting. In other words, the generated code must support the trapping mode. You can see this by compiling a simple test program and look at the machine code listing with

/fpe:0 first, then /fpe:3. There are no fwait instructions in the /fpe:3 code. Even if you replace the default run-time exception handler with your own handler, you may still want to compile with /fpe:0 to generate code that supports trapping.

#### Floating-Point Exceptions in Fortran DLL Applications

In a DLL, there is no main Fortran program (unless you have written your main program in Fortran), so there is no automatic calling of run-time routines to initialize the environment. Even if you select <code>/fpe:0</code>, there is nothing that causes the run-time system to unmask traps in the hardware so you won't see traps. You will continue to see the hardware generated default IEEE results (Nan's, Infinities, and and denormalized numbers in your computations). The generated code will still do its part by supplying the fwait instructions, and so on, but unless the traps are unmasked somehow, no traps will occur. You can use <code>SETCONTROLFPQQ</code> or <code>FOR\_SET\_FPE</code> to unmask traps in the <code>floating-point</code> control word.

There is also no default exception handling in a DLL. The main application that calls the DLL must provide this, or the code in the DLL must provide something when it is called. Since underflow processing (fixup to 0, and so on) is done by the default Fortran run-time system handler, a DLL won't have that feature automatically.

A typical strategy is to compile with /fpe:0, but only unmask floating divide-by-zero, floating overflow, and floating invalid traps in the <u>floating-point control word</u>. By leaving floating-point underflow traps masked, the hardware will continue to provide gradual underflow, but other floating-point exceptions will generate traps, which the user then handles as desired.

#### Floating-Point Exceptions in Fortran Windows Applications

In a Fortran Windows application, the situation is similar to a Fortran DLL application. You define a WinMain routine in your Fortran code as the main entry point for your application. The Fortran run-time system does not define the main

routine as it does in a Fortran console or Fortran QuickWin (or Standard Graphics) application. Your code is not protected by the default handler and the default run-time initialization routines are not called.

# **Understanding IEEE Floating-point Standard**

# **Overview: Understanding IEEE Floating-point Standard**

This version of Intel® Compiler uses a close approximation to the IEEE floating-point standard (ANSI/IEEE Std 754-1985, *IEEE Standard for Binary Floating-Point Arithmetic*, 1985). This standard is common to many microcomputer-based systems due to the availability of fast processors that implement the required characteristics.

This section outlines the characteristics of the standard and its implementation for Intel Compiler. Except as noted, the description includes both the IEEE standard and the Intel Compiler implementation.

# **Floating-point Formats**

The IEEE Standard 754 specifies values and requirements for floating-point representation (such as base 2). The standard outlines requirements for two formats: basic and extended, and for two word-lengths within each format: single and double.

Intel Fortran supports single-precision format (REAL(4)) and double-precision format (REAL(8)) floating-point numbers. At some levels of optimization, some floating-point numbers are stored in register file format (which equals 1 bit sign, 15 bit exponent, 64 bits of significand rounded to 53 bits), rather than being stored in IEEE single or double precision format. This can affect the values produced by some computations. The compiler option -fp-model (Linux\* and Mac OS\* X) or /fp (Windows\*) can control how floating-point expressions are evaluated, thus leading to more predictable results.

#### **Limitations of Numeric Conversion**

The Intel® Fortran floating-point conversion solution is not expected to fulfill all floating-point conversion needs.

For instance, data fields in record structure variables (specified in a STRUCTURE statement) and data components of derived types (TYPE statement) are not converted. When they are later examined as separate fields by the program, they will remain in the binary format they were stored in on disk, unless the program is modified. With EQUIVALENCE statements, the data type of the variable named in the I/O statement is used.

If a program reads an I/O record containing multiple format floating-point fields into a single variable (such as an array) instead of their respective variables, the fields will not be converted. When they are later examined as separate fields by the program, they will remain in the binary format they were stored in on disk, unless the program is modified.

Conversions of the following file structure types are not supported:

- Binary data (FORM= 'BINARY')
- Formatted data (FORM='FORMATTED')
- Unformatted data (FORM='UNFORMATTED') written by Microsoft\* Fortran
   PowerStation or by Intel Fortran with the /fpscomp:ioformat compiler
   option in effect.

## Special Values

The following lists special values that the Intel® Compiler supports and provides a brief description.

# Signed zero

Intel Fortran treats zero as signed by default. The sign of zero is the same as the sign of a nonzero number. If you use the intrinsic function SIGN with zero as the second argument, the sign of the zero will be transferred. Comparisons, however, consider +0 to be equal to -0. A signed zero is useful in certain numerical analysis algorithms, but in most applications the sign of zero is invisible.

#### **Denormalized numbers**

Denormalized numbers (denormals) fill the gap between the smallest positive normalized number and the smallest negative number. Otherwise only (+/-) 0 occurs in that interval. Denormalized numbers extend the range of computable results by allowing for gradual underflow. Systems based on the IA-32 architecture support a Denormal Operand status flag, which, when set, means at least one of the input operands to a floating-point operation is a denormal. The Underflow status flag is set when a number loses precision and becomes a denormal.

### Signed infinity

Infinities are the result of arithmetic in the limiting case of operands with arbitrarily large magnitude. They provide a way to continue when an overflow occurs. The sign of an infinity is simply the sign you obtain for a finite number in the same operation as the finite number approaches an infinite value. By retrieving the status flags, you can differentiate between an infinity that results from an overflow and one that results from division by zero. Intel® Compiler treats infinity as signed by default. The output value of infinity is Infinity or -Infinity.

#### Not a Number

Not a Number (NaN) results from an invalid operation. For instance 0/0 and SQRT(-1) result in NaN. In general, an operation involving a NaN produces another NaN. Because the fraction of a NaN is unspecified, there are many possible NaNs. Intel® Compiler treats all NaNs identically, but provides two different types:

- Signaling NAN, which has an initial fraction bit of 0 (zero). This usually raises an invalid exception when used in an operation.
- Quiet NaN, which has an initial fraction bit of 1.

The floating-point hardware changes a signaling NAN into a quiet NAN during many arithmetic operations, including the assignment operation. An invalid exception may be raised but the resulting floating-point value will be a quiet NAN.

Fortran binary and unformatted input and output do not change the internal representations of the values as they are handled. Therefore, signaling and quiet NANs may be read into real data and output to files in binary form.

The output value of NaN is NaN.

# **Representing Floating-point Numbers**

## Floating-point Representation

The Fortran numeric environment is flexible, which helps make Fortran a strong language for intensive numerical calculations. The Fortran standard purposely leaves the precision of numeric quantities and the method of rounding numeric results unspecified. This allows Fortran to operate efficiently for diverse applications on diverse systems.

Computations on real numbers may not yield what you expect. This happens because the hardware must represent numbers in a finite number of bits.

There are several effects of using finite floating-point numbers. The hardware is not able to represent every real number exactly, but must approximate exact representations by rounding or truncating to finite length. In addition, some numbers lie outside the range of representation of the maximum and minimum exponents and can result in calculations that underflow and overflow. As an example of one consequence, finite precision produces many numbers that, although non-zero, behave in addition as zero.

You can minimize the effects of finite representation with programming techniques; for example, by not using floating-point numbers in LOGICAL comparisons or by giving them a tolerance (for example, IF (ABS(x-10.0) <= 0.001)), and by not attempting to combine or compare numbers that differ by more than the number of significant bits.

Floating-point numbers approximate real numbers with a finite number of bits. The bits are calculated as shown in the following formula. The representation is binary, so the base is 2. The bits  $b_n$  represent binary digits (0 or 1). The precision P is the number of bits in the nonexponential part of the number (the significand),

and *E* is the exponent. With these parameters, binary floating-point numbers approximate real numbers with the values:

$$(-1)$$
s  $b_0$ .  $b_1b_2$  ...  $b_{P-1}$  x  $2^E$ 

where s is 0 or 1 (+ or - ), and  $E_{min} \le E \le E_{max}$ 

The following table gives the standard values for these parameters for single, double, and quad (extended precision) formats and the resulting bit widths for the sign, the exponent, and the full number.

Parameters for IEEE\* Floating-Point Formats

Parameter	Single	Double	Quad or Extended Precision (IEEE_X)*
Sign width in bits	1	1	1
P	24	53	113
$E_{max}$	+127	+1023	+16383
$E_{min}$	-126	-1022	-16382
Exponent bias	+127	+1023	+16383
Exponent width in bits	8	11	15
Format width in bits	32	64	128

<sup>\*</sup> This type is emulated in software.

The actual number of bits needed to represent the precisions 24, 53, and 113 is therefore 23, 52, and 112, respectively, because  $b_0$  is chosen to be 1 implicitly. A *bias* is added to all exponents so that only positive integer exponents occur. This expedites comparisons of exponent values. The stored exponent is actually: e = E + bias

See Also

Native IEEE Floating-Point Representations
Rounding Errors

# **Retrieving Parameters of Numeric Representations**

Intel Fortran includes several intrinsic functions that return details about the numeric representation. These are listed in the following table and described fully in the *Language Reference*.

#### **Functions that Return Numeric Parameters**

Name	Description	Argument/Function Type
DIGITS	DIGITS(x). Returns number of significant digits for data of the same type as x.	x: Integer or Real result: INTEGER(4)
EPSILON	EPSILON( <i>x</i> ). Returns the smallest positive number that when added to one produces a number greater than one for data of the same type as <i>x</i> .	x: Real result: same type as x
EXPONENT	EXPONENT( <i>x</i> ). Returns the exponent part of the representation of <i>x</i> .	x: Real result: INTEGER(4)
FRACTION	FRACTION( <i>x</i> ). Returns the fractional part (significand) of the representation of <i>x</i> .	x: Real result: same type as x
HUGE	HUGE(x). Returns largest number that can be represented by data of type x.	x: Integer or Real result: same type as x.
MAXEXPONENT	MAXEXPONENT( <i>x</i> ). Returns the largest positive decimal exponent for data of the same type as <i>x</i> .	x: Real result: INTEGER(4)
MINEXPONENT	MINEXPONENT( <i>x</i> ). Returns the largest negative decimal exponent for data of the same type as <i>x</i> .	

Name	Description	Argument/Function Type	
NEAREST	NEAREST( <i>x</i> , <i>s</i> ). Returns the nearest different machine representable number to <i>x</i> in the direction of the sign of <i>s</i> .	x: Real s: Real and not zero result: same type as x.	
PRECISION	PRECISION( <i>x</i> ). Returns the number of significant digits for data of the same type as <i>x</i> .	•	
RADIX	RADIX(x). Returns the base for data of the same type as x.	x: Integer or Real result: INTEGER(4)	
RANGE	RANGE(x). Returns the decimal exponent range for data of the same type as x.	x: Integer, Real or Complex result: INTEGER(4)	
RRSPACING	RRSPACING( <i>x</i> ). Returns the reciprocal of the relative spacing of numbers near <i>x</i> .		
SCALE	SCALE( <i>x</i> , <i>i</i> ). Multiplies <i>x</i> by 2 raised to the power of <i>i</i> .	<ul><li>x: Real</li><li>i: Integer</li><li>result: same type as x</li></ul>	
SET_EXPONENT	SET_EXPONENT( <i>x</i> , <i>i</i> ). Returns a number whose fractional part is <i>x</i> and whose exponential part is <i>i</i> .	<ul><li>x: Real</li><li>i: Integer</li><li>result: same type as x</li></ul>	
SPACING	SPACING( <i>x</i> ). Returns the absolute spacing of numbers near <i>x</i> .	<ul><li>x: Real</li><li>result: same type as x</li></ul>	
TINY	TINY(x). Returns smallest positive number that can be represented by data of type x.	x: Real result: same type as x	

#### Native IEEE Floating-point Representation

Native IEEE\* Floating-point Representations Overview

The REAL(4) (IEEE\* S\_floating), REAL(8) (IEEE T\_floating), and REAL(16) (IEEE-style X\_floating) formats are stored in standard little endian IEEE binary floating-point notation. (See IEEE Standard 754 for additional information about IEEE binary floating point notation.) COMPLEX() formats use a pair of REAL values to denote the real and imaginary parts of the data.

All floating-point formats represent fractions in sign-magnitude notation, with the binary radix point to the right of the most-significant bit. Fractions are assumed to be normalized, and therefore the most-significant bit is not stored (this is called "hidden bit normalization"). This bit is assumed to be 1 unless the exponent is 0. If the exponent equals 0, then the value represented is denormalized (subnormal) or plus or minus zero.

Intrinsic REAL kinds are 4 (single precision), 8 (double precision), and 16 (extended precision), such as REAL(KIND=4) for single-precision floating-point data. Intrinsic COMPLEX kinds are also 4 (single precision), 8 (double precision), and 16 (extended precision).

To obtain the kind of a variable, use the KIND intrinsic function. You can also use a size specifier, such as REAL\*4, but be aware this is an extension to the Fortran 95 standard.

If you omit certain compiler options, the default sizes for REAL and COMPLEX data declarations are as follows:

- For REAL data declarations without a kind parameter (or size specifier), the default size is REAL (KIND=4) (same as REAL\*4).
- For COMPLEX data declarations without a kind parameter (or size specifier),
   the default data size is COMPLEX (KIND=4) (same as COMPLEX\*8).

To control the size of all REAL or COMPLEX declarations without a kind parameter, use the /real\_size:64 (or /4R8) or /real\_size:128 (or /4R16) options; the default is /real\_size:32 (or /4R4).

You can explicitly declare the length of a REAL or a COMPLEX declaration using a kind parameter, or specify DOUBLE PRECISION or DOUBLE COMPLEX. To control the size of all DOUBLE PRECISION and DOUBLE COMPLEX declarations, use the /double\_size:128 (or /Qautodouble) option; the default is /double\_size:64.

#### REAL(KIND=4) (REAL) Representation

REAL(4) (same as REAL(KIND=4)) data occupies 4 contiguous bytes stored in IEEE S\_floating format. Bits are labeled from the right, 0 through 31, as shown below.

REAL(4) Floating-Point Data Representation

31	30	23 22		0
SIGN	EXPONE	NT	FRACTION	:А

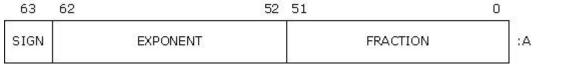
The form of REAL(4) data is sign magnitude, with bit 31 the sign bit (0 for positive numbers, 1 for negative numbers), bits 30:23 a binary exponent in excess 127 notation, and bits 22:0 a normalized 24-bit fraction including the redundant most-significant fraction bit not represented.

The value of data is in the approximate range: 1.17549435E-38 (normalized) to 3.40282347E38. The IEEE denormalized (subnormal) limit is 1.40129846E-45. The precision is approximately one part in 2\*\*23; typically 7 decimal digits.

#### REAL(KIND=8) (DOUBLE PRECISION) Representation

REAL(8) (same as REAL(KIND=8)) data occupies 8 contiguous bytes stored in IEEE T\_floating format. Bits are labeled from the right, 0 through 63, as shown below.

REAL(8) Floating-Point Data Representation



The form of REAL(8) data is sign magnitude, with bit 63 the sign bit (0 for positive numbers, 1 for negative numbers), bits 62:52 a binary exponent in excess 1023

notation, and bits 51:0 a normalized 53-bit fraction including the redundant mostsignificant fraction bit not represented.

The value of data is in the approximate range: 2.2250738585072013D-308 (normalized) to 1.7976931348623158D308. The IEEE denormalized (subnormal) limit is 4.94065645841246544D-324. The precision is approximately one part in 2\*\*52; typically 15 decimal digits.

#### REAL(KIND=16) Representation

REAL(16) (same as REAL(KIND=16)) data occupies 16 contiguous bytes stored in IEEE-style X\_floating format. Bits are labeled from the right, 0 through 127, as shown below.

127	126		112	111		Ö	
SIGN		EXPONENT			FRACTION		:А

The form of REAL(16) data is sign magnitude, with bit 127 the sign bit (0 for positive numbers, 1 for negative numbers), bits 126:112 a binary exponent in excess 16383 notation, and bits 111:0 a normalized 113-bit fraction including the redundant most-significant fraction bit not represented.

The value of data is in the approximate range:

6.4751751194380251109244389582276465524996Q-4966 to

1.189731495357231765085759326628007016196477Q4932. Unlike other floating-point formats, there is little if any performance penalty from using denormalized extended-precision numbers. This is because accessing denormalized REAL (KIND=16) numbers does not result in an arithmetic trap (the extended-precision format is emulated in software). The smallest normalized number is 3.362103143112093506262677817321753Q-4932.

The precision is approximately one part in 2\*\*112 or typically 33 decimal digits.

#### COMPLEX(KIND=4) (COMPLEX) Representation

COMPLEX(4) (same as COMPLEX(KIND=4) and COMPLEX\*8) data is 8 contiguous bytes containing a pair of REAL(4) values stored in IEEE S\_floating format. The low-order 4 bytes contain REAL(4) data that represents the real part

of the complex number. The high-order 4 bytes contain REAL(4) data that represents the imaginary part of the complex number, as shown below. COMPLEX(4) Floating-Point Data Representation

	31	30 23	22	0
REAL PART	SIGN	EXPONENT	FRACTION	:A
IMAGINARY PART	SIGN	EXPONENT	FRACTION	:A+4

The limits and underflow characteristics for REAL (4) apply to the two separate real and imaginary parts of a COMPLEX(4) number. Like REAL(4) numbers, the sign bit representation is 0 (zero) for positive numbers and 1 for negative numbers.

#### COMPLEX(KIND=8) (DOUBLE COMPLEX) Representation

COMPLEX(8) (same as COMPLEX(KIND=8) and COMPLEX\*16) data is 16 contiguous bytes containing a pair of REAL(8) values stored in IEEE T\_floating format. The low-order 8 bytes contain REAL(8) data that represents the real part of the complex data. The high-order 8 bytes contain REAL(8) data that represents the imaginary part of the complex data, as shown below. COMPLEX(8) Floating-Point Data Representation

	63	62 52	51	0
REAL PART	SIGN	EXPONENT	FRACTION	:A
IMAGINARY PART	SIGN	EXPONENT	FRACTION	:A+8

The limits and underflow characteristics for REAL(8) apply to the two separate real and imaginary parts of a COMPLEX(8) number. Like REAL(8) numbers, the sign bit representation is 0 (zero) for positive numbers and 1 for negative numbers.

#### COMPLEX(KIND=16) Representation

COMPLEX(16) (same as COMPLEX(KIND=16) or COMPLEX\*32) data is 32 contiguous bytes containing a pair of REAL(16) values stored in IEEE-style

X\_floating format. The low-order 16 bytes contain REAL(16) data that represents the real part of the complex data. The high-order 16 bytes contain REAL(8) data that represents the imaginary part of the complex data, as shown below.

	127	126 112	111	0
REAL PART	SIGN	EXPONENT	FRACTION	:A
IMAGINARY PART	SIGN	EXPONENT	FRACTION	:A+ 16

The limits and underflow characteristics for <u>REAL(16)</u> apply to the two separate real and imaginary parts of a COMPLEX(16) number. Like REAL(16) numbers, the sign bit representation is 0 (zero) for positive numbers and 1 for negative numbers.

# **Handling Exceptions and Errors**

#### **Loss of Precision Errors**

If a real number is not exactly one of the representable floating-point numbers, then the nearest floating-point number must represent it. The rounding error is the difference between the exact real number and its nearest floating-point representation. If the rounding error is non-zero, the rounded floating-point number is called *inexact*.

Normally, calculations proceed when an inexact value results. Almost any floating-point operation can produce an inexact result. The rounding mode (round up, round down, round nearest, truncate) is determined by the floating-point control word.

If an arithmetic operation results in a floating-point number that cannot be represented in a specific data type, the operation may produce a special value: signed zero, signed infinity, NaN, or a denormal. Numbers that have been rounded to an exactly representable floating-point number also result in a special value. Special-value results are a limiting case of the arithmetic operation involved. Special values can propagate through your arithmetic operations without causing your program to fail, and often provide usable results.

If an arithmetic operation results in an exception, the operation can cause an underflow or overflow:

- Underflow occurs when an arithmetic result is too small for the math
  processor to handle. Depending on the setting of the /fpe compiler option,
  underflows are set to zero (they are usually harmless) or they are left as is
  (denormalized).
- Overflow occurs when an arithmetic result is too large for the math processor
  to handle. Overflows are more serious than underflows, and may indicate an
  error in the formulation of a problem (for example, unintended exponentiation
  of a large number by a large number). Overflows generally produce an
  appropriately signed infinity value. (This depends on the rounding mode as
  per the IEEE standard.)

An arithmetic operation can also throw the following exceptions: divide-by-zero exception, an invalid exception, and an inexact exception.

You can select how exceptions are handled by setting the floating-point control word.

On Windows\* systems, you can select how exceptions are handled by setting the floating-point control word.

See Also

#### **Special Values**

# **Rounding Errors**

Although the rounding error for one real number might be acceptably small in your calculations, at least two problems can arise because of it. If you test for exact equality between what you consider to be two exact numbers, the rounding error of either or both floating-point representations of those numbers may prevent a successful comparison and produce spurious results. Also, when you calculate with floating-point numbers the rounding errors may accumulate to a meaningful loss of numerical significance.

Carefully consider the numerics of your solution to minimize rounding errors or their effects. You might benefit from using double-precision arithmetic or restructuring your algorithm, or both. For instance, if your calculations involve arrays of linear data items, you might reduce the loss of numerical significance by subtracting the mean value of each array from each array element and by normalizing each element of such an array to the standard deviation of the array elements.

The following code segment can execute differently on various systems and produce varying results for n, x, and s. It also produces different results if you use the -fp-model source (Linux\* and Mac OS\* X) or /fp:source (Windows\*; systems based on the IA-32 architecture only), or -fp-model fast (Linux and Mac OS X) or /fp:fast (Windows) compiler options. Rounding error accumulates in x because the floating-point representation of 0.2 is inexact, then accumulates in s, and affects the final value for n:

```
INTEGER n
REAL s, x
n = 0
s = 0.0
x = 0.0
1 n = n + 1
x = x + 0.2
s = s + x
IF ( x .LE. 10. ) GOTO 1 ! Will you get 51 cycles?
WRITE(*,*) 'n = ', n, '; x = ', x, '; s = ', s
```

This example illustrates a common coding problem: carrying a floating-point variable through many successive cycles and then using it to perform an IF test. This process is common in numerical integration. There are several remedies. You can compute x and s as multiples of an integer index, for example, replacing the statement that increments x with x = n \* 0.2 to avoid round-off accumulation. You might test for completion on the integer index, such as IF ( $n \le 50$ ) GOTO 1, or use a DO loop, such as DO n = 1,51. If you must test on the real variable that is being cycled, use a realistic tolerance, such as IF ( $x \le 10.001$ ). Floating-point arithmetic does not always obey the standard rules of algebra exactly. Addition is not precisely associative when round-off errors are considered. You can use parentheses to express the exact evaluation you require to compute a correct, accurate answer. This is recommended when you

specify optimization for your generated code, since associativity may otherwise be unpredictable.

The expressions (x + y) + z and x + (y + z) can give unequal results in some cases.

The compiler uses the default rounding mode (round-to-nearest) during compilation. The compiler performs more compile-time operations that eliminate runtime operations as the optimization level increases. If you set rounding mode to a different setting (other than round-to-nearest), that rounding mode is used only if that computation is performed at runtime. If you want to force computations to be performed at runtime, use the <code>-fp-model strict</code> (Linux or MacOS) or <code>/fp:strict</code> (Windows) option.

See Also

## **OPT-NAME**

# **ULPs, Relative Error, and Machine Epsilon**

ULP, Relative Error, and Machine Epsilon are terms that describe the magnitude of rounding error. A floating-point approximation to a real constant or to a computed result may err by as much as 1/2 unit in the last place (the  $b_{P-1}$  bit). The abbreviation ULP represents the measure "unit in the last place." Another measure of the rounding error uses the relative error, which is the difference between the exact number and its approximation divided by the exact number. The relative error that corresponds to 1/2 ULP is bounded by:

The upper bound  $EPS = 2^{-P}$ , the machine epsilon, is commonly used in discussions of rounding errors because it expresses the smallest floating-point number that you can add to 1.0 with a result that does not round to 1.0.