

# Recommended Intel Compiler Debugging Options

## Category: Program Development Tools

### Commonly Used Options for Debugging

- O0  
Disables optimizations. Default is `-O2`
- g  
Produces symbolic debug information in object file (implies `-O0` when another optimization option is not explicitly set)
- traceback  
Tells the compiler to generate extra information in the object file to provide source file traceback information when a severe error occurs at runtime.  
Specifying `-traceback` will increase the size of the executable program, but has no impact on runtime execution speeds.
- check all  
Checks for all runtime failures.  
Fortran only.
- check bounds  
Alternate syntax: `-CB`. Generates code to perform runtime checks on array subscript and character substring expressions.  
Fortran only.  
Once the program is debugged, omit this option to reduce executable program size and slightly improve runtime performance.
- check uninit  
Checks for uninitialized scalar variables without the *SAVE* attribute.  
Fortran only.
- check-uninit  
Enables runtime checking for uninitialized variables. If a variable is read before it is written, a runtime error routine will be called. Runtime checking of undefined variables is only implemented on local, scalar variables. It is not implemented on dynamically allocated variables, extern variables or static variables. It is not implemented on structs, classes, unions or arrays.  
C/C++ only.
- ftrapuv  
Traps uninitialized variables by setting 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 `-g`.
- debug all

Enables debug information and control output of enhanced debug information. To use this option, you must also specify the `-g` option.

`-gen-interfaces`

`-warn interfaces`

Tells the compiler to generate an interface block for each routine in a source file; the interface block is then checked with `-warn interfaces`.

## Options for Handling Floating-Point Exceptions

`-fpe{0/1/3}`

Allows some control over floating-point exception (divide by zero, overflow, invalid operation, underflow, denormalized number, positive infinity, negative infinity or a NaN) handling for the main program at runtime.

Fortran only.

`-fpe0`: underflow gives 0.0; abort on other IEEE exceptions

`-fpe3`: produce NaN, signed infinities, and denormal results

Default is `-fpe3` with which all floating-point exceptions are disabled and floating-point underflow is gradual, unless you explicitly specify a compiler option that enables flush-to-zero. Use of `-fpe3` on IA-64 systems such as Columbia will slow runtime performance.

`-fpe-all={0/1/3}`

Allows some control over floating-point exception handling for each routine in a program at runtime. Also sets `-assume ieee_fpe_flags`. Default is

`-fpe-all=3`.

Fortran only.

`-assume ieee_fpe_flags`

Tells the compiler to save floating-point exception and status flags on routine entry and restore them on routine exit. This option can slow runtime performance.

Fortran only.

`-ftz`

Flushes denormal results to zero when the application is in the gradual underflow mode. This option has effect only when compiling the main program. It may improve performance if the denormal values are not critical to your application's behavior. For IA-64 systems (such as Columbia) `-O3` sets `-ftz`. For Intel 64 systems (such as Pleiades), every optimization option `O` level, except `-O0`, sets `-ftz`.

## Options for Handling Floating-Point Precision

`-mp`

Enables improved floating-point consistency during calculations. This option limits floating-point optimizations and maintains declared precision. `-mp1` restricts floating-point precision to be closer to declared precision. It has some impact on speed, but less than the impact of `-mp`.

`-fp-model precise`

Tells the compiler to strictly adhere to 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.

`-fp-model 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.

`-fp-speculation=off`

Disables speculation of floating-point operations. Default is

**`-fp-speculation=fast`**

`-pc{64/80}`

For Intel EM64 only. 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 `-pc [n]` option. `-pc 64` sets internal FPU precision to 53-bit significand. `-pc 80` is the default and it sets internal FPU precision to 64-bit significand.

---

Article ID: 92

Last updated: 03 Aug, 2012

Computing at NAS -> Porting & Developing Applications -> Program Development Tools -> Recommended Intel Compiler Debugging Options

<http://www.nas.nasa.gov/hecc/support/kb/entry/92/?ajax=1>