VMS Software

VSI Fortran V8.6-001 for OpenVMS x86-64

Release Notes

Publication Date: November 2024

Operating System: VSI OpenVMS x86-64 Version 9.2-1 or higher

VMS Software, Inc. (VSI)
Boston, Massachusetts, USA

VSI Fortran V8.6-001 for OpenVMS x86-64 Release Notes

VMS Software

1. Release Notes

1.1. Overview

VSI Fortran V8.6-001 for OpenVMS x86-64 Systems runs on VSI OpenVMS x86-64 V9.2-1 or higher. VSI OpenVMS V9.2-3 is strongly recommended.

VSI Fortran on OpenVMS x86-64 is based on VSI Fortran for OpenVMS I64. There may be platform-specific features from OpenVMS Alpha and OpenVMS I64 that may not be supported.

The image identification for the Fortran compiler is:

```
F90 V8.6-001
```

The FORTRAN/VERSION string is:

```
VSI Fortran x86-64 V8.6-001 (GEM 50xxx) on OpenVMS x86_64 V9.2-3
```

1.2. Getting Help And Reporting Problems

Please report problems or offer feedback using the VSI Support Portal.

You can also send comments, questions and suggestions about the VSI Fortran product to `<info@vmssoftware.com>`. Note that these addresses are for informational inquiries and is not a formal support channel.

1.3. Corrections Since The Last Release

- Various math functions (for example, SQRT) now will signal appropriate errors for invalid operations. The returned error code may be slightly different than Alpha or Itanium due to x86 architecture differences. OpenVMS V9.2-3 is required for all errors to be appropriately signalled.
- Improved the online HELP to list all the new predefined symbols for x86 systems.
- Added debug support for variables and parameters access by descriptor. This requires either OpenVMS V9.2-3 or the DEBUG V2 update kit on OpenVMS V9.2-2.
- Fixed the incorrect code when passing a function as an argument to a multi-entry routine.
- Fixed the incorrect code for assigning into an EQUIVALENCE variable.
- Several COMPLEX math operations would return the wrong value.

  Old

  ```
     ( 4.0,3.0 ) ** I = (  7.00,  0.00)
  ```

  New

  ```
     ( 4.0,3.0 ) ** I = (  7.00,  24.00)
  ```

- Using VAX float COMPLEX numbers could crash the compiler. For example:

  ```
  assert error: expression = Ty && "Invalid GetElementPtrInst indices...

          END
  ........^
  %F90-F-FATAL, **Internal compiler error: abort signal raised**
  ```

- The compiler was generating invalid debugger information for structure members which resulted in debug failures or wrong values displayed.

  ```
  DBG> ex struct_1.member_2
  %DEBUG-E-NOACCESSR, no read access to address 2000000000001FF0
  ```

- Allocating arrays in 64-bit address space would lead to run-time errors.

  ```
  program test
  !DEC$ ATTRIBUTES ADDRESS64 :: ttt , i
  real (kind = 8) , allocatable :: ttt( : )
  integer i

  allocate( ttt( 10000 ) )
  ttt( 10000 ) = 8888.0
  write(*,*) ttt( 10000 )
  end
  $ f90 test
  $ link test
  $ run test
  %FOR-F-INVREALLOC, allocatable array is already allocated
  ```

- The length for a COMMON block was computed incorrectly for many programs.

- Various bug fixes for incorrect optimization.

1.4. Changes From OpenVMS Alpha and OpenVMS I64

1.4.1. Incompatible Change For Assigned GOTO Statements

OpenVMS x86-64 places code in 64-bit address space by default. Due to this change, assigning a label to INTEGER*4 variable will cause truncation of a 64-bit pointer to a 32-bit pointer and using **`GOTO`** may cause unexpected results or an access violation error. Change the program to use an INTEGER*8 variable or link with the **`/SEGMENT=CODE=P0`** qualifier which will cause the code segments to be place in 32-bit address space. The compiler will now issue a warning message for using **`ASSIGN`** with an INTEGER*4 variable.

```
$ type assigngoto.for
        integer num

        assign 200 to num
        goto num

200     print *,'At 200'
        end
$ fort assigngoto

        assign 200 to num
.....................^
%F90-W-WARNING, Storing label address to 32-bit integer may cause runtime errors.
at line number 3 in file DKA300:[DIR]ASSIGNGOTO.FOR;1
```

1.4.2. VAX D_floating May Produce Different Results Than I64

VAX D_floating values may produce slightly different results compared to Itanium. This is due to compiler-generated conversions to G_floating and then back to D_floating which removes 3 bits of mantissa. This is similar to Alpha where D_floating would always convert to G_floating before operations. On x86-64, this behavior is acceptable.

1.5. Known Issues

- Some conversions between VAX and IEEE are incorrect with VSI OpenVMS V9.2-1. These problems are fixed in the VSI OpenVMS V9.2-2 release.
- REAL*8 exponentiation does not signal correct errors for invalid operations.
- Debug support is not fully implemented yet, and the debugger may not fully understand Fortran datatypes.
- Quadruple precision floating point (REAL*16, REAL(KIND=16), COMPLEX*16) is currently not supported. This will be addressed in a future release of the compiler along with changes to various system libraries.
- The **`/SEPARATE_COMPILATION`** qualifier is currently ignored. A single Fortran source file is create a single object module.
- The **`/MACHINE_CODE`** qualifier is ignored. As a workaround, you can use the **`ANALYZE/OBJECT/DISASSEMBLE`** to show the generated code and line numbers.

1.6. Known Restrictions

The SCA support with the **`/ANALYSIS_DATA`** qualifier is limited. The Fortran 95 compiler only produces basic SCA information about modules. It does not include information about variables and other symbols. This is a permanent restriction.

1.7. Features Missing From Documentation

IA64, _IA64_ , X86, And _X86_64_ predefine symbols are implemented.

1.8. Floating-Point Arithmetic

- IEEE is the default floating-point datatype (which means, the default is **`/FLOAT=IEEE_FLOAT`**).

  VSI Fortran for OpenVMS Alpha Systems defaults to the VAX G_float floating-point format (**`/FLOAT=G_FLOAT`**). On OpenVMS I64 or OpenVMS x86-64 systems, however, there is no hardware support for floating-point representations other than IEEE. Instead, the compiler supports VAX floating-point formats by generating run-time code which converts operands to IEEE format, performs the needed arithmetic operations, and then converts the IEEE result back to the appropriate VAX format. Depending on the application, this may impose significant additional run-time overhead and some loss of accuracy compared to performing the operations in hardware on an Alpha.

  This software support for the VAX formats is an important functional compatibility requirement for certain applications that need to deal with on- disk binary floating-point data, but its use should be strongly discouraged.

  If possible, users should use **`/FLOAT=IEEE_FLOAT`** (the default) for the highest performance and accuracy.

  Note that the changed **`/FLOAT`** default will have implications for the use of **`/CONVERT=NATIVE`** (the default). This switch causes unformatted data to remain unconverted on input, on the assumption that it matches the prevailing floating-point datatype.

Files written from Fortran applications built with **`/FLOAT=G_FLOAT/CONVERT=NATIVE`** (the default) on Alpha can be read by Integrity server applications built with **`/FLOAT=G_FLOAT/CONVERT=NATIVE`** or **`/FLOAT=IEEE/CONVERT=VAXG`**.

- The **`/IEEE_MODE`** qualifier defaults to **`/IEEE_MODE=DENORM_RESULTS`**. This differs from Alpha, where the default is **`/IEEE_MODE=FAST`**. Despite the name, **`/IEEE_MODE=FAST`** does not have a significant effect on run-time performance on Integrity or x86-64 servers (or on Alpha processors from EV6 onward).
- On Integrity or x86-64 servers, users will have to pick one compile-time **`/FLOAT`** value and one compile-time **`/IEEE_MODE`** value and stick with it for the whole of their application. This is because mixed-mode applications will not (in general) work on OpenVMS I64 or OpenVMS x86-64 systems as a result of architectural differences in the hardware. This is a change from OpenVMS Alpha systems, where mixed-mode applications work. In particular, per-routine/per-file/per-library settings of a mode will not work.
- Exception handlers will be entered with the floating- point mode in effect at the time the exception was raised, not the mode with which the handler was compiled.

2. VSI FORTRAN Documentation and Online Information

The VSI Fortran documentation set can be found online at https://docs.vmssoftware.com/