

# C Language Compatibility Guide

The Vector Fabrics C compiler supports the ANSI C89 standard with its accompanying include files. The most frequently used extensions towards C99 are also supported. Full support of C99 is expected in a later release.

In order to confirm whether your code complies with ANSI C89, you can run the gcc compiler on your local machine with the following options:

```
-ansi  
-std=c89  
-pedantic-errors
```

There are some limitations to the coverage this will provide, as outlined in the [gcc documentation](#). Various other commercial tools may also be available to provide compliance testing.

## Language Support

### Data Types

Vector Fabrics' C compiler supports the standard C scalar datatypes with sizes as in the following table:

type	size
char	8 bits
short	16 bits
int	32 bits
long	32 bits
long long	64 bits
float	32 bits
double	64 bits
long double	<not supported>
(pointer)	32 bits

The integer types (char, short, int, long) can be prefixed with signed or unsigned. They are all signed by default.

Note that in general, for embedded software, it is recommended to use datatypes with explicitly defined bit-size and signedness such as `uint8_t` or `int32_t`. These types are available from the "stdint.h" include file, as of the [C99 standard](#).

### Type and storage qualifiers

The `extern`, `const` and `static` qualifiers are supported according to their C semantics.

The `volatile` type qualifier suppresses some compiler optimizations to ensure memory access consistency. In its current release, the compiler does not fully honor this specification. This has no functional consequences for the current version of the vfAnalyst product, as this is a single-threaded execution model only.

The `restrict` qualifier is supported to allow extra compiler optimizations.

### Predefined macro names

The compiler framework does not use any non-standard defines.

### Pragmas

The Vector Fabrics tools do not make use of any pragmas. Existing pragmas may be recognized by the front-end compiler, but haven't been tested by Vector Fabrics. They may affect the analysis results (for example, loop-unrolling pragmas would result in loops that are no longer present in the analyzed code), but will not cause any tool instability.

### Known limitations

- Inline assembly is not supported.
- MMX and SSE are not supported.
- Function pointers are not supported.
- Passing `struct` constructs by value to variable argument functions is not supported.

## Standard libraries and include files

### C89

The C89 standard header files are supported with the following exceptions:

- `locale.h`
- `wchar.h`
- `iso646.h`
- `wctype.h`
- `signal.h`
- `setjmp.h`

Supported standard header files have the following exceptions:

- for `stdlib.h`, the following are not supported:

- atexit, qsort and bsearch
- wide character functions: wctombs, wctomb
- for stdarg.h, struct constructs passed by value to variable argument functions are not supported

### C99

The C99 standard header files are supported with the following exceptions:

- complex.h
- math.h (misses C99 functions such as sinf and sinl)
- tgmath.h

We are continuously increasing the support for more C header files. We plan to provide most header files from the C99 standard in the coming months. Please provide us with [feedback](#) on your future support needs.

## Run-time model

### Endianness

The C compiler creates an executable with special support for run-time program analysis. This executable runs with a [little-endian memory model](#).

### Alignment

In composite datatypes (struct) individual scalar elements are “naturally aligned”; that is, each one is aligned to a memory address that is a multiple of its size.

## References

- [Overview on the C language](#)
- Home of the [ISO/IEC C language working group](#)
- The [latest C99 working document](#)