# Adding Fundamental Type for Short Float


| | |
|---|---|
| Committee: | ISO/IEC JTC1 SC22 WG14 |
| Document Number: | **N2016** |
| Committee: | ISO/IEC JTC1 SC22 WG21 EWG Evolution |
| Document Number: | **P0192R1** |
| Date: | 2016-02-14 |



| | | |
|---|---|---|
| Authors: | Boris Fomitchev, Sergei Nikolaev, Olivier Giroux, Lawrence Crowl | |
| Reply to: | Boris Fomitchev | boris@stlport.com |
| | Sergei Nikolaev | me@cvmlib.com |
| | Olivier Giroux | ogiroux@nvidia.com |
| | Lawrence Crowl | Lawrence@Crowl.org |


# Contents

# 1 Abstract

We propose adding a new fundamental type, `short float`: a floating-point type of unspecified length, shorter or equal to `float`. The goal is to provide language option to represent platform-specific floating-point values shorter than `float` [1].

# 2 Motivation

One may wonder: why would a programming language need yet another floating point type after so many years of doing just fine without it? Apparently, the times are (a-)changing. Small binary floating-point representation demand and support are becoming more and more common rapidly. Efficient support for hardware that use the new formats becomes mission critical for major software products.

## 2.1 Application use is growing

Application areas include computer graphics, image representation and machine learning. For example, a 16-bit floating-point number better represents the dynamic range of images than 16-bit or even 32-bit integers. A 16-bit floating-point number adequately handles human perceptual range. In 2012, Adobe has defined new HDR DNG image file format that most commonly uses 16-bit floats. [1] This gives those 16-bits much more dynamic range than a traditional file stored as 16 or 32-bit integer data. Both Photoshop and Lightroom, as well as every professional camera produced in 2015 make use of it.

## 2.2 Software support is growing

- OpenGL has `GL_HALF_FLOAT` format since OpenGL3.0. [2]
- The OpenEXR software distribution includes `Half`, a C++ class for manipulating half float values almost as if they were a built-in C++ data type. [3]
- NVIDIA's CUDA 7.5 platform header `cuda_fp16.h` defines the `half` and `half2` data types and defines `__half2float()` and `__float2half()` for conversion between that and `float`. [4]

[1] Usually, this will be "binary16" from IEEE 754-2008 and ISO/IEC 60559:2011

- The gcc compiler provides an `__fp16` native data type extension for ARM. Values of that type are promoted to `float` for computation. [5]
- The LLVM compiler has a 16-bit floating-point type called `half`. [6]

### 2.3 Hardware support is growing

- NVIDIA defined the half data type in the Cg language [7], released in early 2002, and was the first to implement 16-bit floating point in silicon, with the GeForce FX [8], released in late 2002.
- Intel provides instructions for converting between 16-bit and 32-bit floats and C-level intrinsics to use them. [9]
- ARM provides support as an optional extension to the VFPv3 architecture. [10]

Defining standards for 16-bit float math already exist. The IEEE 754 standard defined 'binary16' 16-bit float in 2008. [11] ISO/IEC 60559 ratified that standard in 2011.

However, support for just the IEEE 'binary16' format does not cover existing use cases.

- OpenGL provides 11-bit and 10-bit float channels in `GL_R11F_G11F_B10F` and 14-bit float in `GL_RGB9_E5`.
- Arm provides a 16-bit floating-point format that differs from IEEE 'binary16' by dropping support for NaN and Infinity and then extending the range of values.
- The TI MSP430X architecture provides a 20-bit word-addressed machine. Short floating-point support on that machine would naturally use a 20-bit format.

There is more than enough evidence of the need for a fundamental and platform-defined short floating-point type.

## 3 Existing Solutions

### 3.1 C

There is a draft technical specification of floating-point types extensions in C WG 14 N1896. [12] Proposed solution embeds the length in type names, literal suffixes, etc: `_Float16`. The specification is general in that it can handle a wide range of lengths. However, it only permits an exact size.

### 3.2 C++

The closest related preliminary paper, P0102R0 proposes `stdint`-like support for integers and floats with the sizes specified as template parameters. [13] While it provides better flexibility in size, with 'least' and 'fast' specifications, than does the C TS, P0102R0 only provides for IEEE-conforming implementations. It still does not provide a fundamental type or means to specify "the platform's short float type".

So, both C and C++ lack support for platform-defined short float size and/or possible divergence from IEEE specifications.

## 4 Proposed Solution

We propose adding a new fundamental type, `short float` – for a float type of unspecified (platform defined) bit size, shorter or equal to `float`. Language needs `short float` to represent "shorter than float" math that may be natively available on the platform. This name looks intuitive via `short int` analogy. Most important, it does not introduce any new keywords. We'd be happy to outline few possible alternatives, but we can't think of any that would not require new keywords, would not break any code and still be semantically sound. Also, even if we would consider adding one, it would have to start with `_` and an uppercase in order to not break existing code. It won't even look right in a sentence. Say, we try to use `half`, by `double` example. We'd end up with: `_Half_float`, `float`, `double` and `long double`.

### 4.1 Implementation Options

As of storage and bit-layout for a short float number, we would expect most implementations to follow IEEE 754-2008 [11] half-precision floating point number format. On platform that do not provide any advantages of using shorter float, short float may be implemented as storage-only type, like `__fp16` on gcc/ARM today. For example, it can be stored in 'binary16' format in memory (occupying less bytes than float), converted to native 32-bit float on read from memory, operated on using native 32-bit float math operations and converted back to 'binary16' on store to memory. Or, the platform may choose to not take any advantages of `short float` and represent it using `float` in both memory and registers.

## 5 Impact on the Standards

This section contains brief, though (we believe) complete in scope, overview of changes that would be introduced into C and C++ standard with the proposed addition of `short float`. Should the addition be approved, additional separate documents with exact wording shall be submitted to both 1/SC 22/WG and SC22 WG21.

## 5.1 Lexical Conventions (C and C++)

Adding new fundamental type involves adding literal suffixes specifying floating literals of `short float` type. We suggest the following suffixes: "`sf`" and "SF", for example: `1.2sf` or `1.23e-1SF`. [2]

## 5.2 Declarations (C and C++)

`short float` should be added to the definitions of declarations for fundamental types.

## 5.3 Conversions (C and C++)

Conversion from `short float` to wider floating type are lossless; all short float numbers are exactly representable as wider floats. Conversion from a wider floating type to `short float` possibly may not preserve the float's value exactly — in which case it is rounded to the nearest representable short float using usual rules.

## 5.4 Promotion (C and C++)

`short float` should be promoted to `double`, just like `float` is.

## 5.5 C Library

- ⟨`stdio.h`⟩: For `scanf` format, length prefix "`h`" can be used — natural fit to specifier grammar. For `printf`, promotion will do all the work, no additions necessary.
- ⟨`math.h`⟩: It is to be determined if new are `short float` versions of computational functions defined in ⟨`math.h`⟩ would be required. `sf` function length suffix can be used if those functions added. Implementations should be allowed to implement them as trivial wrappers around functions for float type, using conversions.
- ⟨`float.h`⟩: should follow WG14 TS 18661-3. Compiler-defined `short_float_t` can be added.
- Possible related extension would be to add ⟨`stdfloat.h`⟩ following ⟨`stdint.h`⟩ example, defining floating point types with fixed sizes, present on the platform, e.g. `float16_t`, `float32_t`, `float64_t`. `float16_t` may be defined via short float if it has 16 bit size.

[2] "`s`" and "`h`" suffixes appeared to conflict with new C++ suffixes from ⟨`chrono`⟩

## 5.6 C++ Library

- ⟨cstdfloat⟩: like other C++ C Library headers, should follow ⟨stdfloat.h⟩
- ⟨cmath⟩: overloads for short float should be added
- ⟨limits⟩: numeric_limits⟨short float⟩ should be added
- ⟨numerics⟩: complex⟨short float⟩ should be added

## 5.7 Compatibility

No compatibility issues adding short float — it's fully backward compatible and no C/C++ code can possibly be broken. Some additions proposed in 5.5 may break some C code (e.g. float16_t being contradictory defined by user — such a case would be very unusual). No C++ compatibility issues with 5.5 as those would be in std:: namespace.

# 6 Acknowledgements

Authors would like to thank members of SG6 and and EWG groups of C++ Standard Committee, for consideration of the initial revision [14] of this paper, constructive suggestions and unanimously favorable reception. Also, we appreciate 22/WG 14 being open to consider this proposal in the timeframe that may give a chance to coordinate possible closely related C and C++ Standard changes, to the benefits of users of both languages.

# References

[1] Adobe, "Digital Negative (DNG) Specification", Version 1.4.0.0, 2012 http://wwwimages.adobe.com/content/dam/.../photoshop/pdfs/dng_spec_1.4.0.0.pdf

[2] Segal, Akeley, "The OpenGL (R) Graphics System: A Specification (Version 3.0 – August 11, 2008)" https://www.opengl.org/registry/doc/glspec30.20080811.pdf

[3] Kainz, Bogart, Stanczyk, 'Technical Introduction to OpenEXR', p.18, "The HALF Data Type" http://openexr.com/TechnicalIntroduction.pdf

[4] Mark Harris (NVIDIA), "New Features in CUDA 7.5: 16-bit Floating Point (FP16) Data", 2015 https://devblogs.nvidia.com/parallelforall/new-features-cuda-7-5/

[5] GNU Compilers Manual, 6.12 "Half-Precision Floating Point" https://gcc.gnu.org/onlinedocs/gcc/Half-Precision.html

[6] "LLVM Language Reference Manual', sec. "Floating Point Types" http://llvm.org/releases/3.3/docs/LangRef.html#t-floating

[7] NVIDIA, "Cg Language Specification", sec. "Types"
http://http.developer.nvidia.com/Cg/Cg_language.html

[8] Lal Shimpi, Anand, "NVIDIA Introduces GeForce FX (NV30)", 2002
http://www.anandtech.com/show/1034

[9] Patrick Konsor (Intel), "Performance Benefits of Half Precision Floats"
https://software.intel.com/en-us/articles/performance-benefits-of-half-precision-floats

[10] ARM, "Half-precision floating-point number support"
http://infocenter.arm.com/help/topic/com.arm.doc.dui0205j/CIHGAECI.html

[11] IEEE Computer Society, "IEEE Standard for Floating-Point Arithmetic" IEEE Std 754TM-2008 http://www.csee.umbc.edu/ tsimo1/CMSC455/IEEE-754-2008.pdf

[12] ISO/IEC, "Interchange and extended types" published as ISO/IEC JTC 1/SC 22/WG 14 N1896 http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1896.pdf

[13] Lawrence Crowl, "C++ Parametric Number Type Aliases", ISO/IEC JTC1 SC22 WG21 P0102R0 http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/p0102r0.html

[14] Fomitchev, Nikolaev, Giroux, "Adding Fundamental Type for Short Float", ISO/IEC JTC1 SC22 WG21 P0192R0
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2016/p0192r0.pdf

[15] "Working Draft , Programming Languages – C++", ISO/IEC N4567
http://open-std.org/JTC1/SC22/WG21/docs/papers/2015/n4567.pdf

[16] "Programming languages – C, Committee Draft – April 12, 2011", ISO/IEC 9899:201x
http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf

[17] ISO, "Floating-Point arithmetic", ISO/IEC/IEEE 60559:2011
http://www.iso.org/iso/catalogue_detail.htm?csnumber=57469