Proposal for C2X
WG14 N2560

Title:	FP hex formatting precision
Author, affiliation:	C FP group
Date:	2020-08-20
Proposal category:	Editorial
Reference:	N2478

The **`fprintf`** specification for **`a`**,**`A`** style formatting in 7.21.6 #8 in the current C2X draft (N2478) begins:

> **`a`**,**`A`** A **`double`** argument representing a floating-point number is converted in the style [**`-`**]**`0x`***h*`.`*hhhh***`p`**±*d*, where there is one hexadecimal digit (which is nonzero if the argument is a normalized floating-point number and is otherwise unspecified) before the decimal-point character295) …
>
> 295)Binary implementations can choose the hexadecimal digit to the left of the decimal-point character so that subsequent digits align to nibble (4-bit) boundaries.

Because of the implementation flexibility for the most significant hexadecimal digit, the numerical value of the result might differ among implementations. It would be helpful if the footnote pointed this out. The following suggested change notes the problem, gives an example, and recommends avoiding small precisions in order to get consistent numerical results.

Suggested changes:

1. Change footnote 295 in 7.21.6 #8:

> 295)Binary implementations can choose the hexadecimal digit to the left of the decimal-point character so that subsequent digits align to nibble (4-bit) boundaries. This implementation choice affects numerical values printed with a precision *P* that is insufficient to represent all values exactly. Implementations with different conventions about the most significant hexadecimal digit will round at different places, affecting the numerical value of the hexadecimal result. For example, possible printed output for the code

```
#include <stdio.h>
…
double x = 123.0;
printf("%.1a", x);
```

> include "0x1.fp+6" and "0xf.6p+3" whose numerical values are 124 and 123,

respectively. Portable code seeking identical numerical results on different platforms should avoid precisions P that require rounding.