The recommended practice in 7.23.6.1 (\texttt{fprintf}) recommends correct rounding up to a threshold of $M$ decimal digits for the result, with a looser specification for more than $M$ digits. With this looser specification, increasing the number of output digits could produce a less accurate result: a conversion to $S$ digits could be less accurate than the conversion of the same input to $R$ digits where $M \leq R < S$. This issue was raised to CFP by Vincent Lefevre:

> Let's take an example: $M = 6$, $\approx = 1.2345678$, and rounding to nearest. 
> If the number of significant decimal digits is 6, then the RP says 
> that the correctly rounded value 1.23457 should be output. 
> If the number of significant decimal digits is 7, then one considers 
> $L = 1.23456$ and $U = 1.23457$. According to the RP, 1.234560 is one of 
> the possible recommended outputs, since $1.23456 \leq 1.234560 \leq 1.23457$. 
> Conclusion: By increasing the number of output digits, one has 
> decreased the accuracy! 
> And this can be the case for any number of decimal digits greater 
> than $M$. 
> IMHO, for rounding to nearest on more than $M$ digits, there should 
> be an additional requirement: the error should not be larger than 
> the one for $M$ digits. With this rule, it is still possible to use 
> the correctly rounded value on $M$ digits and pad with zeros. 
> Note that this is a recommendation: if the error is slightly larger 
> than recommended because the algorithm attempts to round correctly 
> for almost all cases, this is not an issue.

The same issue is in 7.31.2.1 (\texttt{fwprintf}).

The following suggested changes address this issue.
Suggested changes (to N3219):

In 7.23.6.1 #13 and in 7.31.2.1 #13, change

the value of the resultant decimal string $D$ should satisfy $L \leq D \leq U$, with the extra stipulations that the error should have a correct sign for the current rounding direction and that increasing the number of decimal digits for the result should not decrease the accuracy of the result.