Adding the new SI prefixes

Document #:	P2734R0
Date:	November 30, 2022
Project:	Programming Language C++ Library Group

Reply-to: Marc Mutz <marc.mutz@hotmail.com>

Abstract

We propose to add the missing SI prefixes `quecto` (10^{-30}), `ronto` (10^{-27}), as well as `ronna` (10^{27}) and `quetta` (10^{30}) to the `<ratio>` header.

1 Motivation and Scope

The General Conference on Weights and Measures (CGPM), at its 27th meeting in November 2022, decided [CGPM 2022 Resolution 3]

> ... to add to the list of SI prefixes to be used for multiples and submultiples of units the following prefixes:

Multiplying factor	Name	Symbol
10^{27}	`ronna`	R
10^{-27}	`ronto`	r
10^{30}	`quetta`	Q
10^{-30}	`quecto`	q

This decision directly affects [**ratio.syn**] and [**ratio.si**], which contain `ratio` typedefs for *each* SI prefix. If the list of SI prefixes grows (it last did in 1991), the corresponding list of `ratio` typedefs needs to follow suit.

2 Implementability & Impact on the Standard

The multiplying factors denominated by the new SI prefixes cannot be represented as a `ratio` when `intmax_t` is 64-bit. It is a property they share with the existing prefixes `yocto`, `zepto`, `zetta` and `yotta`, which are therefore optional in the current IS.

Platforms that can represent 10^{24} (`yotta`) in `intmax_t` can also likely represent 10^{30} (`quetta`) in it (if `intmax_t` is 128-bit ($10^{30} < 2^{100}$)). Even if `intmax_t` is an 80- or 96-bit entity, the problem, from a wording point of view, remains the same: Non-representable typedefs are not required to be provided.

In particular, this proposal is *orthogonal* to potential[1] proposals that wish to guarantee availability of `yotta` etc on all platforms, e.g. by replacing the use of `intmax_t` in the `ratio` interface with a larger extended integer type, an option that may or may not become available were [LWG3828] to be resolved (at the time of writing, it isn't).

Such proposals, however, would be much more demanding on committee time and require LEWG involvement, so this proposal steers clear of such desires and stays within the existing wording for `yotta` etc to deliver the missing SI prefixes with as little effort as possible.

3 Proposed Wording

The following is relative to [N4917]:

- In [**version.syn**], add a row

```
#define __cpp_lib_ratio              YYYYMML // also in <ratio>
```

- Change [**ratio.syn**] as indicated:

```
   // [ratio.si], convenience SI typedefs
+  using quecto = ratio<1, 1'000'000'000'000'000'000'000'000'000'000>;  // see below
+  using ronto  = ratio<1,     1'000'000'000'000'000'000'000'000'000>;  // see below
   using yocto  = ratio<1,         1'000'000'000'000'000'000'000'000>;  // see below
   using zepto  = ratio<1,             1'000'000'000'000'000'000'000>;  // see below
   [...]
   using zetta  = ratio<            1'000'000'000'000'000'000'000, 1>;  // see below
   using yotta  = ratio<          1'000'000'000'000'000'000'000, 1>;  // see below
+  using ronna  = ratio<      1'000'000'000'000'000'000'000'000, 1>;  // see below
+  using quetta = ratio<1'000'000'000'000'000'000'000'000'000, 1>;  // see below
```

  Editorial note: re-indent the whole block to preserve the >–shaped form.

- Change [**ratio.si**] as indicated:

  > For each of the *typedef-name*s `quecto`, `ronto`, `yocto`, `zepto`, `zetta`, ~~and~~ `yotta`, `ronna`, and `quetta`, if both of the constants used in its specification are representable by `intmax_t`, the typedef is defined; if either of the constants is not representable by `intmax_t`, the typedef is not defined.

4 References

[CGPM 2022 Resolution 3] *CGPM/2022-Resolutions (FR-EN)*, Resolution 3, https://www.bipm.org/documents/20126/64811223/Resolutions-2022.pdf/281f3160-fc56-3e63-dbf7-77b76500990f

[N4917] Thomas Köppe, *Working Draft, Standard for Programming Language C++* https://wg21.link/N4917

[LWG3828] GB NB, *Sync intmax_t and uintmax_t with C2x* https://wg21.link/lwg3828

[1] At least, this author is not aware of any such proposals at the time of writing