Some members of the WG21 committee seem to be concerned with the expansion in scope in freestanding environments in C23 due to the changes brought in from the integration of TS 18661. This paper tries to address the concerns brought forward in WG14 reflector messages 19444 and 19445.

Two alternatives are proposed. The first one is intended to reduce the scope of changes brought in by the TS to remove requirements for thread local storage (errno) and modification to "global" state (the floating-point environment, which also has thread storage duration), and does so in a general way. The second alternative is to let implementations choose between having errno or modifying the floating point exception states, or both where possible, and does so by modifying functions that require errno directly. Note that the second alternative does not remove the need to modify the floating-point environment in cases outside certain functions. For example, setting and clearing floating-point exception flags are required.

Note for alternative 1: The practical set of functions in the fenv.h header seem to be only fegetround and fe_dec_getround, but the other query functions were listed as well since they do not modify the floating-point environment either. The reduced list of functions is a possible modification to alternative 1, if chosen.

Alternative 1: In N2596, change Section 4#7 to:

The strictly conforming programs that shall be accepted by a conforming freestanding implementation that defines __STDC_IEC_60559_BFP__ or __STDC_IEC_60559_DFP__ may also use features, without the requirements to set errno (see 7.5) or modify the floating-point environment (see 7.6), in the following:
- all the macros and the following functions in <fenv.h>: fegetexceptflag, fetestexceptflag, fetestexcept, fegetmode, fegetround, fe_dec_getround, fegetenv
- <math.h>
- the floating-point numeric conversion functions (7.22.1) of the standard header <stdlib.h>: strtod, strtof, strtold, strtod32, strtod64, strtod128

All identifiers that are reserved when <stdlib.h> is included in a hosted implementation are reserved when it is included in a freestanding implementation.

Alternative 2: In N2596 change 7.22.1.5#10 to:

The functions return the converted value, if any. If no conversion could be performed, zero is returned. If the correct value overflows and default rounding is in effect (7.12.1), plus or minus HUGE_VAL, HUGE_VALF, or HUGE_VALL is returned (according to the return type and sign of the value); the value of the macro ERANGE is stored in errno if the integer expression math_errhandling & MATH_ERRNO is nonzero; and the "overflow" floating-point exception is raised if the integer expression math_errhandling & MATH_ERREXCEPT is nonzero.

If the result underflows (7.12.1), the functions return a value whose magnitude is no greater than the smallest normalized positive number in the return type; whether errno acquires the value ERANGE if the integer expression math_errhandling & MATH_ERRNO is nonzero is implementation defined. If the integer expression math_errhandling & MATH_ERREXCEPT is nonzero, whether the "underflow" floating-point exception is raised is implementation-defined.

Change 7.22.1.6#7 to:

The strtodN functions return the correctly rounded converted value, if any. If no conversion could be performed, the value of the triple (+1, 0, 0) is returned. If the correct value overflows:

- the value of the macro ERANGE is stored in errno if the integer expression math_errhandling & MATH_ERRNO is nonzero
- the "overflow" floating-point exception is raised if the integer expression math_errhandling & MATH_ERREXCEPT is nonzero

If the result underflows (7.12.1), whether errno acquires the value ERANGE if the integer expression math_errhandling & MATH_ERRNO is nonzero is implementation defined; if the integer expression math_errhandling & MATH_ERREXCEPT is nonzero, whether the "underflow" floating-point exception is raised is implementation-defined.

Change 4#7 as follows:

and the floating-point numeric conversion functions (7.22.1) of the standard header <stdlib.h>