Changes to TS 18661 Part 3
Interchange and extended types

WG 14 N1765
2013-10-02
Part 3 draft N1758

- TS 18661 Part 3 is C support for new IEC 60559 formats
- N1758 updates N1691 discussed in Delft
- Goal: show changes, get input, update for next meeting
Interchange formats

- IEC 60559-2011 introduced a "tower" of interchange formats
- Arbitrarily large widths (32x)
- Precision and range determined by width
- binary16, for GPU data etc.
- For exchange of FP data
- May or may not be arithmetic
Extended formats

• IEC 60559-2011 specifies *extended* formats that extend its basic formats (binary32|64|128 and decimal64|128)
• Have at least a specified precision and range
• For explicit wide evaluation
• Not for data exchange
IEC 60559 format support

• IEC 60559 formats:
  Interchange formats
    – Arithmetic
    – Non-arithmetic
  Extended formats
  *Extendable formats*

• Arithmetic interchange and extended formats fully supported as floating types
• Non-arithmetic interchange formats supported without additional types
• Extendable formats not covered
Type structure additions

real floating types
  standard floating types: float, double, long double
  interchange floating types
    _FloatN
  decimal floating types: _DecimalN
  extended floating types: _FloatNx, _DecimalNx

complex types
  float _Complex, double _Complex, long double _Complex
  _FloatN _Complex, _FloatNx _Complex

Imaginary types
  float _Imaginary, double _Imaginary, long double _Imaginary
  _FloatN _Imaginary, _FloatNx _Imaginary
Type structure unchanged

floating types
  real floating types
  complex types
  imaginary types

real types
  integer types
  real floating types

arithmetic types
  integer types
  floating types
Non-arithmetic interchange formats

• Supported as encodings, not types
• Encodings stored in unsigned char arrays
• Required conversion operations provided by library functions
• Arithmetic interchange formats are supported as encodings and as types
Requirements

• Types are distinct and not compatible
• Requires interchange and extended floating types whose formats must already be supported because of conformance to Part 1 or 2
• Requires support for binary16 format, at least as an encoding (if Part 1 is supported)
• Allows support for other interchange floating types and encodings
• Requires complex (and imaginary) types for supported binary interchange and extended floating types
Example 1

Assume

- Part 1 conformance
- long double has common IEEE 80-bit extended format

<table>
<thead>
<tr>
<th>Required new type</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Float32</td>
<td>32</td>
</tr>
<tr>
<td>_Float64</td>
<td>64</td>
</tr>
<tr>
<td>_Float32x</td>
<td>64 or 80</td>
</tr>
<tr>
<td>_Float64x</td>
<td>80</td>
</tr>
</tbody>
</table>

And complex (and imaginary) types for all of above

Required binary encoding widths: 16, 32, 64
Example 2

Assume

– Part 1 conformance
– long double has IEEE binary128 format

<table>
<thead>
<tr>
<th>Required new type</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Float32</td>
<td>32</td>
</tr>
<tr>
<td>_Float64</td>
<td>64</td>
</tr>
<tr>
<td>_Float128</td>
<td>128</td>
</tr>
<tr>
<td>_Float32x</td>
<td>64 or 128</td>
</tr>
<tr>
<td>_Float64x</td>
<td>128</td>
</tr>
</tbody>
</table>

And complex (and imaginary) types for all of above

Required binary encoding widths: 16, 32, 64, 128
Example 3

Assume

– Part 2 conformance

<table>
<thead>
<tr>
<th>Required type</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Decimal32</td>
<td>32</td>
</tr>
<tr>
<td>_Decimal64</td>
<td>64</td>
</tr>
<tr>
<td>_Decimal128</td>
<td>128</td>
</tr>
<tr>
<td>_Decimal64x</td>
<td>128</td>
</tr>
</tbody>
</table>

Required decimal encoding widths: 32, 64, 128
Encoding functions

For all supported interchange floating types ...
• Encode – type-to-encoding (same format)
• Decode – encoding-to-type (same format)

For all supported IEC 60559 encodings ...
• Encoding-to-encoding conversions
• String-to-encoding conversions
• String-from-encoding conversions

➤ Each decimal type and encoding requires two sets of encoding functions, one for each decimal encoding scheme
Example 4

Assume
   – Part 1 conformance
   – long double has common IEEE 80-bit extended format
   – binary16 supported only as an encoding

To convert binary16 encoding stored in
  unsigned char e16[2];
to
  _Float32 f32;
use
  unsigned char e32[4];
f32encf16(e32, e16);
decodef32(&f32, e32);
TS (re)organization

• Conformance to Part 3 requires conformance to Part 1 or Part 2 (or both)
• Specification is cumulative: C11 (+ TC 1) + Part 1 + Part 2 + Part 3
• Changes in Part 3 are applied to C11 + Part 1 + Part 2
• Part 3 decimal specification generalizes Part 2, so \textit{decimal floating types} include all \texttt{_DecimalN}
• Identifiers controlled by WANT macros listed in header clauses (Page 2 Line 23 -)