

Errata Sheet for IEEE Std 1076.1-1999 IEEE Standard VHDL Analog and Mixed-Signal Extensions

The following conventions are used in this document:

- Text to be struck out or replaced is marked by ~~strike-through~~ font
- New text or replacing text is presented as underlined text
- Explanatory text that is not part of the LRM uses a non-serif font
- Sections of text where only the layout (presentation) changed are introduced by Improved layout:
and reprinted in the new layout (no strike-through or underlining)

Clause 1.1.1.2, last paragraph

The 5th line of this paragraph should also have a change bar.

Clause 3.1.1.1, second paragraph

The predefined type CHARACTER is a character type whose values are the 256 characters of the ISO 8859-1: ~~1997~~ 1987 [B4]² character set.

Clause 3.2.1.2, 4th paragraph

The values of the predefined type ~~types~~ BIT_VECTOR and REAL_VECTOR are one-dimensional arrays of the predefined type BIT and REAL, respectively, indexed by values of the predefined subtype NATURAL:

Clause 3.3.1

Example of a recursive type:

...

```
variable HEAD : LINK := new CELL'(0, null, null);  
variable \NEXT\ : LINK := HEAD.SUCC;
```

Clause 3.5.1, second example

nature ~~thermal~~Thermal **is**

```
Temperature across      -- across type is Temperature  
Heatflow through      -- through type is Heatflow  
Th_ref reference;      -- reference terminal is named Th_ref
```

Clause 3.5.2.2, second example

A mixed record nature

```

nature mixed is
record
    ElJunction: Electrical;
    ThJunction: Thermal;
end record;
nature Mixed is
    record
        ElJunction: Electrical;
        ThJunction: Thermal;
end record;

```

-- *Illegal: Thermal and Electrical do not have the same*
-- *simple nature.*

Clause 4.3.1.1

The first line of the Examples should have a change bar.

Clause 5 Specifications

This clause should start on a new page.

Clause 5.1, 18th paragraph

The single exception to this is the predefined attribute LAST_EVENT ~~LAST_EVENT~~, where the two different attributes are disambiguated by the return type of the attribute.

Clause 5.4

— For a composite quantity Q, an explicit or implicit step limit specification of the form

```
limit limit Q: T with real_expression;
```

...

Example:

```

library IEEE;
use Ieee.Math_real.all;
entity source is
    generic (Amplitude: REAL; Freq: REAL);
    port (quantity Sine: out REAL);
limit Sine: REAL with 0.05/Freq;
end entity source;

architecture Sinusoid of source is
limit Sine: REAL with 0.05/Freq;
begin
    Sine == Amplitude * sin ( Math_2_pi * Freq * NOW );
end architecture Sinusoid;

```

-- ensures that there are at least 20 analog
-- solution points per period of the sine wave

Clause 6.1

A name is said to be a static name if and only if one of the following conditions holds:

- ...
 - The name is an attribute name whose prefix is a static quantity name and whose suffix is the predefined attribute 'ABOVE.

Clause 6.3, Note 1

```
list2 := list1;           -- Access values are copied;
                        -- list1 and list2 now denote the same object.
list2 := list1.\next\;  -- list2 denotes the same object as list1.\next\.
                        -- list1.\next\ is the same as list1.all.\next\.
                        -- An implicit dereference of the access value occurs before the
                        -- "\next\" field is selected.
recobj := list2.all;    -- An explicit dereference is needed here.
recobj := list2.all;    -- An explicit dereference is needed here.
```

Clause 12.6.4, initialization

- 8) The time of the next simulation cycle (which in this case is the first simulation cycle), T_n , is set to 0.0.

Clause 12.6.5.4

NOTE—The definitions above prefixed by the ieee.math_real and ieee_math_complex package names are defined in IEEE Std 1076.2-1996 [B6].

Clause 12.6.5.5

NOTE—The definitions above prefixed by the ieee.math_real and ieee_math_complex package names are defined in IEEE Std 1076.2-1996 [B6].

Clause 12.7

Improved layout:

- If N_0 is that value of type TIME equal to one resolution limit unit then

$$0.0 \leq \text{universal_to_physical_time}(U_1+U_2) - \text{universal_to_physical_time}(U_1) - \text{universal_to_physical_time}(U_2) \leq N_0.$$
- If R_0 is the smallest positive value of type Universal_Time such that

$$R_0 = \text{universal_to_real_time}(U_0+U_1+U_2) - \text{universal_to_real_time}(U_1+U_2) > 0.0,$$
 then

$$0.0 \leq \text{universal_to_real_time}(U_1+U_2) - \text{universal_to_real_time}(U_1) - \text{universal_to_real_time}(U_2) \leq R_0.$$

Clause 13.1

- f) Other special characters

! \$ % @ ? \ ^ ` { } ~ | ¢ £ ¥ ¦ § ¨ © ª « ¬ ® ¯ ° ± ² ³ ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ × ÷ - (soft hyphen)
! \$ % @ ? \ ^ ` { } ~ | ¢ £ ¥ ¦ § ¨ © ª « ¬ ® ¯ ° ± ² ³ ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ × ÷ - (soft hyphen)

Note 4

...

Character	Name
¥	Yen sign
† † † †	Broken bar
§	Paragraph sign, clause sign
¨	Diaeresis
©	Copyright sign
^a	Feminine ordinal indicator
«	Left angle quotation mark
¬	Not sign
-	Soft hyphen *
®	Registered trade mark sign
-	Macron
°	Ring above, degree sign
±	Plus-minus sign
²	superscript two
³	Superscript three
´	Acute accent
µ	Micro sign
¶	Pilcrow sign
•	Middle dot
¸	Cedilla
¹	Superscript one
º	Masculine ordinal indicator
»	Right angle quotation mark
¼	Vulgar fraction one quarter
½	Vulgar fraction one half
¾	Vulgar fraction three quarters
¿	Inverted question mark
×	Multiplication sign
÷	Division sign

Clause 13.3.2

```
extended_identifier ::=
    \graphic_character { graphic_character } \
```

Clause 13.9

The line starting with **component** should not have a change bar.

2—An extended identifier whose sequence of characters inside the leading and trailing backslashes is identical to a reserved word is not a reserved word. For example, `\next\` is a legal (extended) identifier and is not the reserved word **next**.

Clause 14.1

A'LENGTH[(N)]

Result: Number of values in the Nth index range; i.e., if the Nth index range of A is a null range, then the result is 0. Otherwise, the result is the value of $T'POS(A'HIGH(N)) - T'POS(A'LOW(N))$ result is the value of $T'POS(A'HIGH(N)) - T'POS(A'LOW(N)) + 1$, where T is the subtype or subnature of the Nth index of A.

S'LAST_EVENT

Result Type: Type Time.

For a signal S, S'LAST_EVENT returns the smallest value T of type TIME such that S'EVENT = True during any simulation cycle at time $NOW - T$ NOW - T, if such a value exists; otherwise, it returns TIME'HIGH.

S'LAST_EVENT

Result Type: Type Real.

For a signal S, S'LAST_EVENT returns the smallest value T of type REAL such that S'EVENT = True during any simulation cycle at time NOW - T, if such a value exists; otherwise, it returns universal_to_real_time(U) where U is the universal time corresponding to TIME'HIGH.

S'LAST_ACTIVE

For a signal S, S'LAST_ACTIVE returns the smallest value T of type TIME such that S'ACTIVE = True during any simulation cycle at time $NOW - T$ NOW - T, if such value exists; otherwise, it returns TIME'HIGH.

E'PATH_NAME

architecture A of E is

```

signal S: BIT_VECTOR (1 to G);           -- S'PATH_NAME = "e:s"
                                           -- S'INSTANCE_NAME = "e(a):s"
procedure Proc1 (signal sp1: NATURAL; C: out INTEGER) is
                                           -- Proc1'PATH_NAME = "e:proc1:"
                                           -- Proc1'INSTANCE_NAME = "e(a):proc1:"
                                           -- C'PATH_NAME = "e:proc1:c"
                                           -- C'INSTANCE_NAME = "e(a):proc1:c"
variable max: DELAY_LENGTH;           -- max'PATH_NAME = "e:proc1:max"
variable max: DELAY_LENGTH;           -- max'PATH_NAME = "e:proc1:max"
                                           -- max'INSTANCE_NAME =
                                           -- "e(a):proc1:max"

```

begin

...

Q'ZOH(T[,INITIAL_DELAY])

Improved layout:

```

...
if DOMAIN = FREQUENCY_DOMAIN use
  if FREQUENCY = 0.0 use
    zoh == q;
  else
    zoh == q'Delayed(0.5 * t) *
      sin(FREQUENCY*pi_t)/(FREQUENCY*pi_t);
  end use;
elsif DOMAIN'DELAYED = QUIESCENT_DOMAIN use
  if initial_delay = 0.0 use
    zoh == q;
  else
    zoh == 0.0;
  end use;
else
  zoh == s;
end use;
...

```

Q'LTF(NUM, DEN)

Restrictions: The first scalar subelement of DEN must not be 0.0.

Restrictions: The first scalar subelement of DEN must not be 0.0.

Q'ZTF(NUM,DEN,T[,INITIAL_DELAY])

```

use IEEE.math_real.all;
block
  generic (num, den: REAL_VECTOR; t, initial_delay: REAL);
  generic map (num => NUM, den => DEN, t=>T,
    partial_delay=>INITIAL_DELAY);
    initial_delay=>INITIAL_DELAY);
  port (quantity q: in REAL;
    quantity ztf: out REAL);
...

```

S'RAMP[(TRISE [,TFALL])]

```

block
  generic (trise, tfall: REAL);
  generic map (trise => REAL(TRISE), tfall => REAL(TFALL));
  port (signal s: in REAL; quantity sramp: out REAL);
  port map (s => REAL(SELEM), STYPE(sramp) => SRAMP);
  port map (s => REAL(SELEM), STYPE(sramp) => SRAMP);
  signal slope, tt: REAL := 0.0;
...

```

S'SLEW[(RISING_SLOPE [,FALLING_SLOPE])]

block

```

generic (rising_slope, falling_slope: REAL);
generic map (rising_slope => RISING_SLOPE, falling_slope =>
              FALLING_SLOPE);
port (signal s: in REAL; quantity sslew: out REAL);
port map (s => REAL(SELEM), STYPE(sslew) => SSLEW);
port map (s => REAL(SELEM), STYPE(sslew) => SSLEW);
signal slope, tt: REAL := 0.0;

```

begin

...

Improved layout:

```

while s'EVENT loop
  if s > sslew and rising_slope /= REAL'HIGH then
    ttv := (s-sslew)/rising_slope;
    slope <= rising_slope;
  elsif s < sslew and falling_slope /= REAL'LOW then
    ttv := (s-sslew)/falling_slope;
    slope <= falling_slope;
  else
    ttv := 0.0;
  end if;
  tt <= ttv;
  wait on s for ttv;
end loop;

```

Q'SLEW[(MAX_RISING_SLOPE [,MAX_FALLING_SLOPE])]

Suppose that QSLEW is an alias for any scalar subelement of the attribute name, that QTYPE designates its type, that QELEM is an alias for the corresponding scalar subelement of S Q, and that MAX_RISING_SLOPE and MAX_FALLING_SLOPE designate the actual or defaulted values of the parameters with the same names. Then the behavior of each scalar subelement of Q'SLEW is formally described by the value of QSLEW produced by the following equivalent block:

block

```

generic (max_rising_slope, max_falling_slope: REAL);
generic map (max_rising_slope => MAX_RISING_SLOPE,
              max_falling_slope => MAX_FALLING_SLOPE);
port (quantity q: in REAL; quantity qslew: out REAL);
port map (q => REAL(QELEM), QTYPE(qslew) => QSLEW);
port map (q => REAL(QELEM), QTYPE(qslew) => QSLEW);
type slewstate is (falling, none, rising);
signal slewing: slewstate := none;

```

...

6—If S'STABLE(T) is FALSE, then, by definition, for some t where $0 \text{ ns} \leq t \leq T$, S'DELAYED(t) /= S.

6—If S'STABLE(T) is FALSE, then, by definition, for some t where $0 \text{ ns} < t < T$, S'DELAYED(t) /= S.

Clause 14.2

type CHARACTER is (

```

...
C128,   C129,   C130,   C131,   C132,   C133,   C134,   C135,
C136,   C137,   C138,   C139,   C140,   C141,   C142,   C143,
C144,   C145,   C146,   C147,   C148,   C149,   C150,   C151,
C152,   C153,   C154,   C155,   C156,   C157,   C158,   C159,
'!', *  '¡',   'ç',   '£',   '¤',   '¥',   '¦',   '§',
'¨',   '©',   'ª',   «',   '¬',   '­',   '®',   '¯',
'°',   '±',   '²', '₂', '³', '₃',   '´',   'µ',   '¶',   '·', '₆', '₇',
'¸',   '¹', '¹',   'º',   '»',   '¼', '¼', '½', '½', '¾', '¾',   '¿',
'À',   'Á',   'Â',   'Ã',   'Ä',   'Å',   'Æ',   'Ç',
'È',   'É',   'Ê',   'Ë',   'Ì',   'Í',   'Î',   'Ï',
'Ð',   'Ñ',   'Ò',   'Ó',   'Ô',   'Õ',   'Ö',   '×',
'Ø',   'Ù',   'Ú',   'Û',   'Ü',   'Ý',   'Þ',   'ß',
'à',   'á',   'â',   'ã',   'ä',   'å',   'æ',   'ç',
'è',   'é',   'ê',   'ë',   'ì',   'í',   'î',   'ï',
'ð',   'ñ',   'ò',   'ó',   'ô',   'õ',   'ö',   '÷',
'ø',   'ù',   'ú',   'û',   'ü',   'ý',   'þ',   'ÿ');

```

...

The following line should not have a change bar:

```

subtype DELAY_LENGTH is TIME range 0 fs to TIME'HIGH;

```

...

-- A function that returns the current simulation frequency (see 12.8):

-- A function that returns the current simulation frequency (see 12.8):

```

function FREQUENCY return REAL;

```

Clause 15.4, Example

-- Its equivalent statements are

```

type p1_type is

```

```

type p1_type is

```

```

record

```

```

    \Id\: Electrical'Through;

```

```

    \Charge\: REAL;

```

```

    \Ic\: Electrical'Through;

```

```

end record;

```

...

```

variable \Charge'Dot\ : REAL := \Charge'Dot\

```

```

variable \Charge'Dot\ : REAL := \Charge'Dot\;

```

```

begin

```

```

    \Id\ := Is0*(exp((\Vd\ - Rd*\Id\)/Vt) - 1.0);

```

```

— \Charge\ := Tau*\Id\

```

```

— \Ic\ := \Charge'Dot\

```

```

    \Charge\ := Tau*\Id\;

```

```

— \Ic\ := \Charge'Dot\;

```

```

    return (\Id\, \Charge\, \Ic\);

```

```

end function;

```

Annex A

```
extended_identifier ::= \_graphic_character { graphic_character } \      [§ 13.3.2]
...
quantity_declaration ::=                                             [§ 4.3.1.6]
  — | free_quantity_declaration
     | free_quantity_declaration
     | branch_quantity_declaration
     | source_quantity_declaration
...
subtype_indication ::=                                             [§ 4.2]
  — [ resolution_function_name ] type_mark [ constraint ] [ tolerance_aspect ]
  — [ resolution_function_name ] type_mark [ constraint ] [ tolerance_aspect ]
```

Annex B

The following entries should have change bars: B.29, B.65 and B.135.

B.135 implicit signal: Any signal S'Stable(T), S'Quiet(T), S'Delayed, or S'Transaction, or any implicit GUARD signal, or any signal of the form Q'Above(E). A slice or subelement (or slice thereof) of an implicit signal is also an implicit signal. (§12.6.2, §12.6.3, §12.6.4)