

# The LUA-PHYSICAL library

Version 1.0.3

Thomas Jenni

September 10, 2020

## Abstract

`lua-physical` is a pure Lua library, which provides functions and objects for the computation of physical quantities. A physical quantity is the product of a numerical value and a physical unit. The package has been written, to simplify the creation physics problem sets. The package provides units of the SI and the imperial system. Furthermore, an almost complete set of international currencies are supported, however without realtime exchange rates. In order to display the numbers with measurement uncertainties, the package is able to perform gaussian error propagation.

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Dependencies . . . . .	3
<b>2</b>	<b>Loading</b>	<b>3</b>
2.1	License . . . . .	4
<b>3</b>	<b>Usage</b>	<b>6</b>
3.1	Unit conversion . . . . .	6
3.1.1	Temperature Conversion . . . . .	8
3.2	Uncertainty Propagation . . . . .	9
3.3	Mathematical operations . . . . .	11
<b>4</b>	<b>Supported Units</b>	<b>13</b>
4.1	Prefixes . . . . .	13
4.2	Base Units . . . . .	14
4.3	Constants . . . . .	16
4.4	Coherent derived units in the SI . . . . .	17
4.5	Non-SI units accepted for use with the SI . . . . .	18
4.6	Nominal Astronomical Units . . . . .	18
4.7	Other Non-SI units . . . . .	19

4.8	Imperial Units . . . . .	21
4.9	U.S. customary units . . . . .	23
4.10	International Currencies . . . . .	24
4.10.1	Pegged International Currencies . . . . .	32
<b>5</b>	<b>Lua Documentation</b>	<b>36</b>
5.1	physical.Quantity . . . . .	36
5.2	physical.Dimension . . . . .	47
5.3	physical.Unit . . . . .	48
5.4	physical.Number . . . . .	49
<b>6</b>	<b>Change History</b>	<b>58</b>
	<b>Bibliography</b>	<b>59</b>
	<b>Index of Units</b>	<b>60</b>
	<b>Index of Currencies</b>	<b>63</b>
	<b>Index of Lua Classes and Methods</b>	<b>67</b>

# 1 Introduction

The author of this package is a physics teacher at the high school *Kantonsschule Zug*, Switzerland. The main use of this package is to write physics problem sets. Lua<sup>A</sup>T<sub>E</sub>X does make it possible to integrate physical calculations directly. The package has been in use since 2016. Many bugs have been found and fixed. Nevertheless it still is possible, that some were not found yet. Therefore the author recommends not to use this package in industry or science. If one does so, it's the responsibility of the user to check results for plausability. If the user finds some bugs, they can be reported at [github.com](https://github.com).

## 1.1 Dependencies

This is a standalone library. However, it is compatible with the `siunitx` package. The results of calculations can be written to Lua<sup>A</sup>T<sub>E</sub>X directly by calling the `physical.Quantity.tosiunitx()` method. It is recommended to use a macro for this purpose. The preamble in the next section, simplifies the printing of quantities by the `\q{}` macro.

## 2 Loading

This Lua library can be loaded by calling `require("physical")`. The following Lua<sup>A</sup>T<sub>E</sub>X preamble loads the `lua-physical` package and creates a macro `\q` for printing physical quantities.

Listing 1: basic preamble

```
\usepackage{luacode}
\usepackage{siunitx}

% configure siunitx (siunitx package)
\sisetup{
  output-decimal-marker = {.},
  per-mode = symbol,
  separate-uncertainty = true,
  add-decimal-zero = true,
  exponent-product = \cdot,
  round-mode = off
}

% declare the unitless unit (siunitx package)
\DeclareSIUnit\number{}

% load the lua-physical package
\begin{luacode*}
  physical = require("physical")
  N = physical.Number

  N.omitUncertainty = true
\end{luacode*}
```

```

\end{luacode*}

% print a physical quantity
\newcommand{\q}[1]{%
  \directlua{
    tex.print(
      physical.Quantity.tosiunitx(
        #1,
        "scientific-notation=fixed,exponent-to-prefix=false"
      )
    )
  }%
}

% print the unit in scientific notation
\newcommand{\qs}[1]{%
  \directlua{
    tex.print(
      physical.Quantity.tosiunitx(
        #1,
        "scientific-notation=true,exponent-to-prefix=false,
        round-integer-to-decimal=true"
      )
    )
  }%
}

% print the unit of a quantity
\newcommand{\qu}[1]{%
  \directlua{
    tex.print(
      physical.Quantity.tosiunitx(
        #1,
        nil,
        physical.Quantity.SIUNITX_si
      )
    )
  }%
}

```

## 2.1 License

This code is freely distributable under the terms of the MIT license.

Copyright (c) 2020 Thomas Jenni

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### 3 Usage

Given the basic preamble, units can be used in lua code directly. By convention, all units have an underscore in front of them, i.e. meter is `_m`, second is `_s`. All available units are listed in section 4. The following example illustrates the use of this library.

Listing 2: Velocity of a car.

```
\begin{luacode}
  s = 10 * _m
  t = 2 * _s
  v = s/t
\end{luacode}

A car travels  $\text{\q{s}}$  in  $\text{\q{t}}$ . Calculate its velocity.
%
\begin{equation*}
  v=\frac{s}{t} = \frac{\text{\q{s}}}{\text{\q{t}}} = \underline{\text{\q{v}}}
\end{equation*}
```

A car travels 10 m in 2 s. Calculate its velocity.

$$v = \frac{s}{t} = \frac{10 \text{ m}}{2 \text{ s}} = \underline{5.0 \text{ m/s}}$$

In the above listing 2, the variable `s` stands for displacement and has the unit meter `_m`, the variable `t` stands for time and is given in seconds `_s`. If mathematical operations are done on them, new derived quantities are created. In the example problem above, the velocity `v` is calculated by dividing `s` by `t` and has therefore the unit `_m/_s`. By using the macro `\q{}` all quantities can be printed to the Lua<sup>A</sup>T<sub>E</sub>X code directly.

#### 3.1 Unit conversion

It is often the case, that the result of a calculation has to be converted to another unit. Lets assume, that in the problem of listing 2, the velocity should be determined in `_km/_h`.

Listing 3: Velocity of a car.

```
\begin{luacode}
  s = 10 * _m
  t = 2 * _s
  v = (s / t):to(_km/_h)
\end{luacode}
```

A car travels  $s$  in  $t$ . Calculate its velocity in  $\text{km/h}$ .

```
%
\begin{equation*}
v = \frac{s}{t} = \frac{q{s}}{q{t}} = \underline{q{v}}
\end{equation*}
```

A car travels 10 m in 2 s. Calculate its velocity in km/h.

$$v = \frac{s}{t} = \frac{10 \text{ m}}{2 \text{ s}} = \underline{18.0 \text{ km/h}}$$

It is the function `:to()`, which does the unit conversion. This function is available on all physical quantities. If no argument is given, the quantity is converted to base units.

Another example is given in listing 4. The task is to calculate the volume of a cuboid. The length of the edges are given in different units. The volume is calculated by multiplying all three lengths. The result has the unit  $\text{cm mm m}$ . If the unit  $\text{cm}^3$  is preferred, it has to be converted explicitly. At first this looks a bit cumbersome. The reason of this behaviour is, that the software is not able to guess the unit of the result. In many cases, like in the example problem, it's not clear what unit the volume should have. Is it  $\text{m}^3$ ,  $\text{cm}^3$  or  $\text{L}$  (liter) ? The user has to give that conversion explicitly.

Listing 4: Volume of a cuboid.

```
\begin{luacode}
a = 12 * _cm
b = 150 * _mm
c = 1.5 * _m

V = ( a * b * c ):to(_dm^3)
\end{luacode}

Find the volume of a rectangular cuboid with lengths  $q{a}$ ,
 $q{b}$  and  $q{c}$ .
%
\begin{equation*}
V = a \cdot b \cdot c
= q{a} \cdot q{b} \cdot q{c}
= q{V}
= \underline{q{V}}
\end{equation*}
```

Find the volume of a rectangular cuboid with lengths 12 cm, 150 mm and 1.5 m.

$$V = a \cdot b \cdot c = 12 \text{ cm} \cdot 150 \text{ mm} \cdot 1.5 \text{ m} = 27.0 \text{ dm}^3 = \underline{27.0 \text{ dm}^3}$$

### 3.1.1 Temperature Conversion

Most physical units transform linearly. Exceptions are the unit degree Celsius `_degC` and degree Fahrenheit `_degF`. These units are ambiguous and can be interpreted as temperature differences or as an absolute temperatures. In the latter case, the conversion to base units is not a linear, but an affine transformation. This is because degree Celsius and degree Fahrenheit scales have their zero points at different temperatures compared to the unit Kelvin.

By default `_degC` and `_degF` units are temperature differences. If one wants to have it converted absolutely, it has to be done adding / subtracting `_degC_0 = 273.15*_K` or `_degF_0 = (273.15 - 32*(5/9)) * _K`, the zero point temperatures of the scales.

In the following problem, listing 5, the task is to convert temperatures given in the unit degree Celsius and degree Fahrenheit to Kelvin.

Listing 5: Temperature conversion.

```

\begin{luacode}
  theta_1 = 110 * _degC
  T_1 = ( theta_1 + _degC_0 ):to(_K)

  T_2 = 100 * _K
  theta_2 = ( T_2 - _degC_0 ):to(_degC)

  theta_3 = 212 * _degF
  T_3 = ( theta_3 + _degF_0 ):to(_K)

  T_4 = 100 * _K
  theta_4 = ( T_4 - _degF_0 ):to(_degF)

  theta_5 = 100 * _degC
  theta_6 = ( ( theta_5 + _degC_0 ):to(_K) - _degF_0 ):to(_degF)
\end{luacode}

\begin{align*}
\qquad \backslash q{\theta_1} \ \&\backslash mathrel{\backslash widehat{=}} \ \backslash q{T_1} \ \backslash \\
\qquad \% \\
\qquad \backslash q{\theta_2} \ \&\backslash mathrel{\backslash widehat{=}} \ \backslash q{T_2} \ \backslash \\
\qquad \% \\
\qquad \backslash q{\theta_3} \ \&\backslash mathrel{\backslash widehat{=}} \ \backslash q{T_3} \ \backslash \\
\qquad \% \\
\qquad \backslash q{\theta_4} \ \&\backslash mathrel{\backslash widehat{=}} \ \backslash q{T_4} \ \backslash \\
\qquad \% \\
\qquad \backslash q{\theta_5} \ \&\backslash mathrel{\backslash widehat{=}} \ \backslash q{\theta_6} \ \backslash \\
\end{align*}

```



$$\begin{aligned}
110\text{ }^{\circ}\text{C} &\hat{=} 383.15\text{ K} \\
-173.15\text{ }^{\circ}\text{C} &\hat{=} 100\text{ K} \\
212\text{ }^{\circ}\text{F} &\hat{=} 373.15\text{ K} \\
-279.67\text{ }^{\circ}\text{F} &\hat{=} 100\text{ K} \\
100\text{ }^{\circ}\text{C} &\hat{=} 212.0\text{ }^{\circ}\text{F}
\end{aligned}$$

### 3.2 Uncertainty Propagation

The `lua-physical` library supports uncertainty propagation. To create a number with an uncertainty, an instance of `physical.Number` has to be created, see listing 6. It has to be remembered, that `N` is a alias for `physical.Number`. The first argument of the constructor `N(mean, uncertainty)` is the mean value and the second one the uncertainty of the measurement. If the proposed preamble 1 is used, the uncertainty is by default separated from the mean value by a plus-minus sign.

For the uncertainty propagation the gaussian formula

$$\Delta f = \sqrt{\left(\frac{\partial f}{\partial x_1} \cdot \Delta x_1\right)^2 + \dots + \left(\frac{\partial f}{\partial x_n} \cdot \Delta x_n\right)^2}$$

is used. This formula is a good estimation for the uncertainty  $\Delta f$ , if the quantities  $x_1, \dots, x_n$  the function  $f$  depends on, have no correlation. Further, the function  $f$  has to change linear, if quantities  $x_i$  are changed in the range of their uncertainties.

Listing 6: Uncertainty in area calculation.

```

\begin{luacode}
  N.omitUncertainty = false

  a = N(2,0.1) * _m
  b = N(3,0.1) * _m

  A = ( a * b ):to(_m^2)
\end{luacode}

Calculate the area of a rectangle with lengths  $\text{\textbackslash q{a}}$  and  $\text{\textbackslash q{b}}$ .
%
\begin{equation*}
  A = a \cdot b
    = \text{\textbackslash q{a}} \cdot \text{\textbackslash q{b}}
    = \underline{\text{\textbackslash q{A}}}
\end{equation*}

```

Calculate the area of a rectangle with lengths  $(2.00 \pm 0.10) \text{ m}$  and  $(3.00 \pm 0.10) \text{ m}$ .

$$A = a \cdot b = (2.00 \pm 0.10) \text{ m} \cdot (3.00 \pm 0.10) \text{ m} = \underline{(6.0 \pm 0.4) \text{ m}^2}$$

Instead of printing the uncertainties, one can use the uncertainty calculation to provide significant digits and omit it.

In the following problem, listing 7, the task is to find the volume of an ideal gas. Given are pressure  $p$  in `_bar`, amount of substance  $n$  in `_mol` and temperature  $T$  in degree celsius `_degC`. In order to do the calculation, one has to convert  $T$ , which is given as an absolute temperature in degree celsius to the base unit Kelvin first. By setting `N.omitUncertainty = true`, all uncertainties are not printed.

Listing 7: Volume of an ideal gas.

```
\begin{luacode}
  N.omitUncertainty = true
  p = N(1.013,0.0001) * _bar
  n = N(1,0.01) * _mol
  T = N(30,0.1) * _degC

  V = ( n * _R * (T + _degC_0):to(_K) / p ):to(_L)
\end{luacode}

An ideal gas ( $\text{\textbackslash q{n}}$ ) has a pressure of  $\text{\textbackslash q{p}}$  and a temperature of  $\text{\textbackslash q{T}}$ . Calculate the volume of the gas.
%
```

$$V = \frac{n \cdot R \cdot (T + \text{\textbackslash degC\_0})}{p}$$

```
\begin{equation*}
  V = \frac{n \cdot R \cdot (T + \text{\textbackslash degC\_0})}{p}
  = \underline{\text{\textbackslash q{V}}}
\end{equation*}
```

An ideal gas (1.0 mol) has a pressure of 1.013 bar and a temperature of 30 °C. Calculate the volume of the gas.

$$V = \frac{1.0 \text{ mol} \cdot 8.31 \text{ J}/(\text{mol K}) \cdot 303 \text{ K}}{1.013 \text{ bar}} = \underline{25 \text{ L}}$$

This example shows, that the result has only two digits. If more digits are needed, the uncertainties of the given quantities should be smaller.

### 3.3 Mathematical operations

Two physical quantities with identical dimensions can be added or subtracted. The library checks the validity of those operations and throws an error if two addends haven't the same dimensions.

Listing 8: Addition and subtraction

```
l_1 = 1 * _m
l_2 = 2 * _cm
t = 2 * _s

l_1 + t
Error: Cannot add '1* _m' to '2 * _s', because they have different
dimensions.

l_1 + l_2
102.0 * _cm
```

New physical quantities can be created by division and multiplication. As long as no division by zero is made, no errors should occur.

Listing 9: Multiplication and Division

```
l_1 = 1 * _m
l_2 = 2 * _cm

(l_1 * l_2):to(_m^2)
0.02 * _m^2

(l_1 / l_2):to(_1)
50.0 * _1
```

Physical quantities can be exponentiated. The library doesn't check, if the result has units with non integer exponents.

Listing 10: Exponentiation

```
l = 5 * _m
A = l^2

A:to(_m^2)
25.0 * _m^2

A:sqrt()
5.0 * _m

A^0.5
5.0 * _m
```

Exponential functions and the logarithms should have dimensionless arguments. The library throws an error if that's not the case.

Listing 11: Exponential function and logarithm

```
N_0 = 1000 * _1
lambda = Q.log(2)/(2*_h)
t = 50 * _min

N_0 * Q.exp(-lambda * t)
749.15353843834 * _1
```

## 4 Supported Units

All supported units are listed in this section. Subsection 4.2 lists the seven base units of the International System of Units (SI). In subsection 4.3 mathematical and physical constants are defined. The subsection 4.4 contains all coherent derived units from the SI system and 4.5 those which are accepted to use with the SI. The subsection 4.6 lists nominal astronomical units, which are proposed by [4]. Subsection 4.7 lists units, which are common but outside of the SI system. The subsections 4.8 and 4.9 are dedicated to imperial and U.S. customary units. The last subsection 4.10 contains international currencies.

### 4.1 Prefixes

All SI units have prefixed versions, i.e. `_us` microsecond, `_cm` centimeter, `_mN` millinewton, see table 1. Some units of data processing, like `_bit` have prefixes which are powers of 2. They are called binary or IEC prefixes, see table 2 [2, 121].

Prefix	Symbol	Definition	Prefix	Symbol	Definition
yotta	Y	1e24	deci	d	1e-1
zetta	Z	1e21	centi	c	1e-2
exa	E	1e18	milli	m	1e-3
peta	P	1e15	micro	u	1e-6
tera	T	1e12	nano	n	1e-9
giga	G	1e9	pico	p	1e-12
mega	M	1e6	femto	f	1e-15
kilo	k	1e3	atto	a	1e-18
hecto	h	1e2	zepto	z	1e-21
deca	da	1e1	yocto	y	1e-23

Table 1: SI prefixes [2, 121]

Prefix	Symbol	Definition
kibi	Ki	1024
mebi	Mi	1048576
gibi	Gi	1073741824
tebi	Ti	1099511627776
pebi	Pi	1125899906842624
exbi	Ei	1152921504606846976
zebi	Zi	1180591620717411303424
yobi	Yi	1208925819614629174706176

Table 2: IEC prefixes [2, 121]

## 4.2 Base Units

The `lua-physical` library has nine base quantities. These are the seven basis units or basis quantities of the SI system [3] and in addition the base quantity of information `_bit` and of currency `_EUR`. All other quantities are derived from these base units.

Quantity	Unit	Symbol	Dim.	Definition
number <sup>1</sup>	–	<code>_1</code>	1	The dimensionless number one.
time	second	<code>_s</code>	T	The SI unit of time. It is defined by taking the fixed numerical value of the caesium frequency $\Delta\nu_{Cs}$ , the unperturbed ground-state hyperfine transition frequency of the caesium 133 atom, to be 9 192 631 770 when expressed in the unit 1/s.
length	meter	<code>_m</code>	L	The SI unit of length. It is defined by taking the fixed numerical value of the speed of light in vacuum $c$ to be 299 792 458 when expressed in the unit of m/s.

<sup>1</sup>The number one is a unit with dimension zero. Stricly speaking it is not a base unit.

Quantity	Unit	Symbol	Dim.	Definition
mass	kilogram	<code>_kg</code>	M	The SI unit of mass. It is defined by taking the fixed numerical value of the Planck constant $h$ to be $6.626\,070\,15 \cdot 10^{-34}$ when expressed in $\text{m}^2 \text{kg/s}$ .
electric current	ampere	<code>_A</code>	I	The SI unit of electric current. It is defined by taking the fixed numerical value of the elementary charge $e$ to be $1.602\,176\,634 \cdot 10^{-19}$ when expressed in A s.
thermodynamic temperature	kelvin	<code>_K</code>	K <sup>1</sup>	The SI unit of the thermodynamic temperature. It is defined by taking the fixed numerical value of the Boltzmann constant $k_B$ to be $1.380\,649 \cdot 10^{-23}$ when expressed in $\text{kg m}^2/(\text{s}^2 \text{K})$
amount of substance	mole	<code>_mol</code>	N	The SI unit of amount of substance. One mole contains exactly $6.022\,140\,76 \cdot 10^{23}$ elementary entities. This number is the fixed numerical value of the Avogadro constant $N_A$ when expressed in 1/mol.
luminous intensity	candela	<code>_cd</code>	J	The SI unit of luminous intensity in a given direction. It is defined by taking the fixed numerical value of the luminous efficacy of monochromatic radiation of frequency $5.4 \cdot 10^{14} \text{ Hz}$ , $K_{cd}$ , to be 683 when expressed in the unit $\text{cd sr s}^3/(\text{kg m}^2)$ .
information	bit	<code>_bit</code>	B	The smallest amount of information.
currency	euro	<code>_EUR</code>	C	The value of the currency Euro.

Table 3: Base units

<sup>1</sup>The SI symbol for the dimension of temperature is  $\Theta$ , but all symbols of this library consist of roman letters, numbers and underscores only. Therefore the symbol for the dimension of the thermodynamic temperature is the letter K.

### 4.3 Constants

All physical constants are taken from the NIST webpage [1].

Name	Symbol	Definition
pi	<code>_Pi</code>	$3.1415926535897932384626433832795028841971 * \_1$
eulersnumber	<code>_E</code>	$2.7182818284590452353602874713526624977572 * \_1$
speedoflight	<code>_c</code>	$299792458 * \_m/_s$
gravitationalconstant	<code>_Gc</code>	$N(6.67408e-11, 3.1e-15) * \_m^3/(\_kg*_s^2)$
planckconstant	<code>_h_P</code>	$6.62607015e-34 * \_J*_s$
reducedplanckconstant	<code>_h_Pbar</code>	$\_h\_P/(2*_Pi)$
elementarycharge	<code>_e</code>	$1.602176634e-19 * \_C$
vacuumpermeability	<code>_u_0</code>	$4e-7*_Pi * \_N/_A^2$
vacuumpermittivity	<code>_e_0</code>	$1/(\_u_0*_c^2)$
atomicmassunit	<code>_u</code>	$N(1.66053904e-27, 2e-35) * \_kg$
electronmass	<code>_m_e</code>	$N(9.10938356e-31, 1.1e-38) * \_kg$
protonmass	<code>_m_p</code>	$N(1.672621898e-27, 2.1e-35) * \_kg$
neutronmass	<code>_m_n</code>	$N(1.674927471e-27, 2.1e-35) * \_kg$
bohrmagneton	<code>_u_B</code>	$\_e*_h\_Pbar/(2*_m\_e)$
nuclearmagneton	<code>_u_N</code>	$\_e*_h\_Pbar/(2*_m\_p)$
electronmagneticmoment	<code>_u_e</code>	$N(-928.4764620e-26, 5.7e-32) * \_J/_T$
protonmagneticmoment	<code>_u_p</code>	$N(1.4106067873e-26, 9.7e-35) * \_J/_T$
neutronmagneticmoment	<code>_u_n</code>	$N(-0.96623650e-26, 2.3e-26) * \_J/_T$
finestructureconstant	<code>_alpha</code>	$\_u_0*_e^2*_c/(2*_h\_P)$
rydbergconstant	<code>_Ry</code>	$\_alpha^2*_m\_e*_c/(2*_h\_P)$
avogadronumber	<code>_N_A</code>	$6.02214076e23/_mol$
boltzmannconstant	<code>_k_B</code>	$1.380649e-23 * \_J/_K$
molargasconstant	<code>_R</code>	$N(8.3144598, 4.8e-6) * \_J/(\_K*_mol)$
stefanboltzmannconstant	<code>_sigma</code>	$\_Pi^2*_k\_B^4/(60*_h\_Pbar^3*_c^2)$
standardgravity	<code>_g_0</code>	$9.80665 * \_m/_s^2$

Table 4: Physical and mathematical constants



## 4.4 Coherent derived units in the SI

All units in this section are coherent derived units from the SI base units with special names, [2, 118].

Quantity	Unit	Symbol	Definition
Plane Angle <sup>1</sup>	radian	<code>_rad</code>	<code>_1</code>
Solid Angle <sup>2</sup>	steradian	<code>_sr</code>	<code>_rad^2</code>
Frequency	hertz	<code>_Hz</code>	<code>1/_s</code>
Force	newton	<code>_N</code>	<code>_kg*_m/_s^2</code>
Pressure	pascal	<code>_Pa</code>	<code>_N/_m^2</code>
Energy	joule	<code>_J</code>	<code>_N*_m</code>
Power	watt	<code>_W</code>	<code>_J/_s</code>
Electric Charge	coulomb	<code>_C</code>	<code>_A*_s</code>
Electric Potential	volt	<code>_V</code>	<code>_J/_C</code>
Electric Capacitance	farad	<code>_F</code>	<code>_C/_V</code>
Electric Resistance	ohm	<code>_Ohm</code>	<code>_V/_A</code>
Electric Conductance <sup>3</sup>	siemens	<code>_S</code>	<code>_A/_V</code>
Magnetic Flux	weber	<code>_Wb</code>	<code>_V*_s</code>
Magnetic Flux Density	tesla	<code>_T</code>	<code>_Wb/_m^2</code>
Inductance	henry	<code>_H</code>	<code>_Wb/_A</code>
Temperature <sup>4</sup>	celsius	<code>_degC</code>	<code>_K</code>
Luminous Flux	lumen	<code>_lm</code>	<code>_cd*_sr</code>
Illuminance	lux	<code>_lx</code>	<code>_lm/_m^2</code>
Activity	becquerel	<code>_Bq</code>	<code>1/_s</code>
Absorbed Dose	gray	<code>_Gy</code>	<code>_J/_kg</code>
Dose Equivalent	sievert	<code>_Sv</code>	<code>_J/_kg</code>
Catalytic Activity	katal	<code>_kat</code>	<code>_mol/_s</code>

<sup>1</sup>In the SI system, the quantity Plane Angle has the dimension of a number.

<sup>2</sup>In the SI system, the quantity Solid Angle has the dimension of a number.

<sup>3</sup>The unit `_PS` stands for peta siemens and is in conflict with the metric version of the unit horsepower (german *Pferdestärke*). Since the latter is more common than peta siemens, `_PS` is defined to be the metric version of horsepower.

<sup>4</sup>The unit `_degC` is by default interpreted as a temperature difference.

## 4.5 Non-SI units accepted for use with the SI

There are a few units with dimension 1. [2, 124].

Quantity	Unit	Symbol	Definition
Time	minute	<code>_min</code>	$60 * \text{\_s}$
	hour	<code>_h</code>	$60 * \text{\_min}$
	day	<code>_d</code>	$24 * \text{\_h}$
Plane Angle	degree	<code>_deg</code>	$(\text{\_Pi}/180) * \text{\_rad}$
	arcminute	<code>_arcmin</code>	$\text{\_deg}/60$
	arcsecond	<code>_arcsec</code>	$\text{\_arcmin}/60$
Area	hectare	<code>_hectare</code>	$1e4 * \text{\_m}^2$
Volume	liter	<code>_L</code>	$1e-3 * \text{\_m}^3$
Mass	tonne	<code>_t</code>	$1e3 * \text{\_kg}$

## 4.6 Nominal Astronomical Units

The nominal values of solar, terrestrial and jovial quantities are taken from IAU Resolution B3 [4].

Quantity	Unit	Symbol	Definition
Length	<code>nomsolradius</code>	<code>_R_S_nom</code>	$6.957e8 * \text{\_m}$
Irradiance	<code>nomsolirradiance</code>	<code>_S_S_nom</code>	$1361 * \text{\_W}/\text{\_m}^2$
Radiant Flux	<code>nomsolluminosity</code>	<code>_L_S_nom</code>	$3.828e26 * \text{\_W}$
Temperature	<code>nomsolefttemperature</code>	<code>_T_S_nom</code>	$5772 * \text{\_K}$
Mass Parameter	<code>nomsolmassparameter</code>	<code>_GM_S_nom</code>	$1.3271244e20 * \text{\_m}^3 * \text{\_s}^{-2}$
Length	<code>nomterreqradius</code>	<code>_Re_E_nom</code>	$6.3781e6 * \text{\_m}$
Length	<code>nomterrpolarradius</code>	<code>_Rp_E_nom</code>	$6.3568e6 * \text{\_m}$
Mass Parameter	<code>nomterrmassparameter</code>	<code>_GM_E_nom</code>	$3.986004e14 * \text{\_m}^3 * \text{\_s}^{-2}$
Length	<code>nomjoveqradius</code>	<code>_Re_J_nom</code>	$7.1492e7 * \text{\_m}$
Length	<code>nomjovpolarradius</code>	<code>_Rp_J_nom</code>	$6.6854e7 * \text{\_m}$
Mass Parameter	<code>nomjovmassparameter</code>	<code>_GM_J_nom</code>	$1.2668653e17 * \text{\_m}^3 * \text{\_s}^{-2}$

## 4.7 Other Non-SI units

The unit Bel is only available with prefix decibel, because `_B` is the unit byte.

Quantity	Unit	Symbol	Definition
Length	angstrom	<code>_angstrom</code>	$1\text{e-}10 * \text{\_m}$
	fermi	<code>_fermi</code>	$1\text{e-}15 * \text{\_m}$
Time	svedberg	<code>_svedberg</code>	$1\text{e-}13 * \text{\_s}$
	week	<code>_wk</code>	$7 * \text{\_d}$
	year	<code>_a</code>	$365.25 * \text{\_d}$
	astronomicalunit	<code>_au</code>	$149597870700 * \text{\_m}$
	lightsecond	<code>_ls</code>	$\text{\_c} * \text{\_s}$
	lightyear	<code>_ly</code>	$\text{\_c} * \text{\_a}$
	parsec	<code>_pc</code>	$(648000/\text{\_Pi}) * \text{\_au}$
Area	barn	<code>_barn</code>	$1\text{e-}28 * \text{\_m}^2$
	are	<code>_are</code>	$1\text{e}2 * \text{\_m}^2$
Volume	metricteaspoon	<code>_tsp</code>	$5\text{e-}3 * \text{\_L}$
	metrictablespoon	<code>_Tbsp</code>	$3 * \text{\_tsp}$
Plane Angle	gradian	<code>_gon</code>	$(\text{\_Pi}/200) * \text{\_rad}$
	turn	<code>_tr</code>	$2 * \text{\_Pi} * \text{\_rad}$
Solid Angle	spat	<code>_sp</code>	$4 * \text{\_Pi} * \text{\_sr}$
Force	kilopond	<code>_kp</code>	$\text{\_kg} * \text{\_g}_0$
Pressure	bar	<code>_bar</code>	$1\text{e}5 * \text{\_Pa}$
	standardatmosphere	<code>_atm</code>	$101325 * \text{\_Pa}$
	technicalatmosphere	<code>_at</code>	$\text{\_kp}/\text{\_cm}^2$
	millimeterofmercury	<code>_mmHg</code>	$133.322387415 * \text{\_Pa}$
	torr	<code>_Torr</code>	$(101325/760) * \text{\_Pa}$
Energy	thermochemicalcalorie	<code>_cal</code>	$4.184 * \text{\_J}$
	internationalcalorie	<code>_cal_IT</code>	$4.1868 * \text{\_J}$
	gramoftnt	<code>_g_TNT</code>	$1\text{e}3 * \text{\_cal}$
	tonoftnt	<code>_t_TNT</code>	$1\text{e}9 * \text{\_cal}$

Quantity	Unit	Symbol	Definition
	electronvolt	<code>_eV</code>	<code>_e*_V</code>
	wattsecond	<code>_Ws</code>	<code>_W*_s</code>
	watthour	<code>_Wh</code>	<code>_W*_h</code>
Power	voltampere	<code>_VA</code>	<code>_V*_A</code>
Electric Charge	amperesecond	<code>_As</code>	<code>_A*_s</code>
	amperehour	<code>_Ah</code>	<code>_A*_h</code>
Information	nibble	<code>_nibble</code>	<code>4 *_bit</code>
	byte	<code>_B</code>	<code>8 *_bit</code>
Information Transfer Rate	bitpersecond	<code>_bps</code>	<code>_bit/_s</code>
Number	percent	<code>_percent</code>	<code>1e-2 *_1</code>
	permille	<code>_permille</code>	<code>1e-3 *_1</code>
	partspermillion	<code>_ppm</code>	<code>1e-6 *_1</code>
	partsperbillion	<code>_ppb</code>	<code>1e-9 *_1</code>
	partspertrillion	<code>_ppt</code>	<code>1e-12 *_1</code>
	partsperquadrillion	<code>_ppq</code>	<code>1e-15 *_1</code>
	decibel	<code>_dB</code>	<code>_1</code>
Power	metrichorsepower	<code>_PS</code>	<code>75 *_g_0*_kg*_m/_s</code>
Activity	curie	<code>_Ci</code>	<code>3.7e10 *_Bq</code>
Absorbed Dose	rad	<code>_Rad</code>	<code>1e-2 *_Gy</code>
Dose Equivalent	rem	<code>_rem</code>	<code>1e-2 *_Sv</code>
Viscosity	poiseuille	<code>_Pl</code>	<code>_Pa*_s</code>

## 4.8 Imperial Units

Quantity	Unit	Symbol	Definition
Length	inch	<code>_in</code>	$2.54\text{e-}2 * \text{\_m}$
	thou	<code>_th</code>	$1\text{e-}3 * \text{\_in}$
DTP Point <sup>1</sup>	point	<code>_pt</code>	$\text{\_in}/72$
	pica	<code>_pica</code>	$12 * \text{\_pt}$
	hand	<code>_hh</code>	$4 * \text{\_in}$
	foot	<code>_ft</code>	$12 * \text{\_in}$
	yard	<code>_yd</code>	$3 * \text{\_ft}$
	rod	<code>_rd</code>	$5.5 * \text{\_yd}$
	chain	<code>_ch</code>	$4 * \text{\_rd}$
	furlong	<code>_fur</code>	$10 * \text{\_ch}$
	mile	<code>_mi</code>	$8 * \text{\_fur}$
	league	<code>_lea</code>	$3 * \text{\_mi}$
	nauticalmile	<code>_nmi</code>	$1852 * \text{\_m}$
	nauticalleague	<code>_nlea</code>	$3 * \text{\_nmi}$
	cable	<code>_cbl</code>	$0.1 * \text{\_nmi}$
	fathom	<code>_ftm</code>	$6 * \text{\_ft}$
Velocity	knot	<code>_kn</code>	$\text{\_nmi}/\text{\_h}$
Area	acre	<code>_ac</code>	$10 * \text{\_ch}^2$
Volume	gallon	<code>_gal</code>	$4.54609 * \text{\_L}$
	quart	<code>_qt</code>	$\text{\_gal}/4$
	pint	<code>_pint</code>	$\text{\_qt}/2$
	cup	<code>_cup</code>	$\text{\_pint}/2$
	gill	<code>_gi</code>	$\text{\_pint}/4$
	fluidounce	<code>_fl_oz</code>	$\text{\_gi}/5$
	fluid dram	<code>_fl_dr</code>	$\text{\_fl_oz}/8$

---

<sup>1</sup>The desktop publishing point or PostScript point is 1/72 of an international inch.

Quantity	Unit	Symbol	Definition
Mass	grain	<code>_gr</code>	$64.79891 * \text{mg}$
	pound	<code>_lb</code>	$7000 * \text{gr}$
	ounce	<code>_oz</code>	$\text{lb}/16$
	dram	<code>_dr</code>	$\text{lb}/256$
	stone	<code>_st</code>	$14 * \text{lb}$
	quarter	<code>_qtr</code>	$2 * \text{st}$
	hundredweight	<code>_cwt</code>	$4 * \text{qtr}$
	longton	<code>_ton</code>	$20 * \text{cwt}$
	troy pound	<code>_lb_t</code>	$5760 * \text{gr}$
	troy ounce	<code>_oz_t</code>	$\text{lb}_t/12$
	pennyweight	<code>_dwt</code>	$24 * \text{gr}$
	firkin	<code>_fir</code>	$56 * \text{lb}$
Time	sennight	<code>_sen</code>	$7 * \text{d}$
	fortnight	<code>_ftn</code>	$14 * \text{d}$
Temperature <sup>1</sup>	fahrenheit	<code>_degF</code>	$(5/9) * \text{K}$
Force	poundforce	<code>_lbf</code>	$\text{lb} * \text{g}_0$
	poundal	<code>_pdl</code>	$\text{lb} * \text{ft} / \text{s}^2$
Mass	slug	<code>_slug</code>	$\text{lbf} * \text{s}^2 / \text{ft}$
Pressure	poundforcepersquareinch	<code>_psi</code>	$\text{lbf} / \text{in}^2$
Energy, Torque	thchembritishthermalunit	<code>_BTU</code>	$(1897.83047608/1.8) * \text{J}$
Energy, Torque	intbritishthermalunit	<code>_BTU_it</code>	$1055.05585262 * \text{J}$
Power	horsepower	<code>_hp</code>	$33000 * \text{ft} * \text{lbf} / \text{min}$

---

<sup>1</sup>The unit `_degF` is by default interpreted as a temperature difference.

## 4.9 U.S. customary units

In the U.S., the length units are bound to the meter differently than in the imperial system. The following definitions are taken from [https://en.wikipedia.org/wiki/United\\_States\\_customary\\_units](https://en.wikipedia.org/wiki/United_States_customary_units).

Quantity	Unit	Symbol	Definition
Length	ussurveyinch	_in_US	_m/39.37
	ussurveyhand	_hh_US	4 * _in_US
	ussurveyfoot	_ft_US	3 * _hh_US
	ussurveylink	_li_US	0.66 * _ft_US
	ussurveyyard	_yd_US	3 * _ft_US
	ussurveyrod	_rd_US	5.5 * _yd_US
	ussurveychain	_ch_US	4 * _rd_US
	ussurveyfurlong	_fur_US	10 * _ch_US
	ussurveymile	_mi_US	8 * _fur_US
	ussurveyleague	_lea_US	3 * _mi_US
	ussurveyfathom	_ftm_US	72 * _in_US
	ussurveycable	_cbl_US	120 * _ftm_US
Area	ussurveyacre	_ac_US	_ch_US * _fur_US
Volume	usgallon	_gal_US	231 * _in^3
	usquart	_qt_US	_gal_US/4
	uspint	_pint_US	_qt_US/2
	uscup	_cup_US	_pint_US/2
	usgill	_gi_US	_pint_US/4
	usfluidounce	_fl_oz_US	_gi_US/4
	ustablespoon	_Tbsp_US	_fl_oz_US/2
	usteaspoon	_tsp_US	_Tbsp_US/3
	usfluid dram	_fl_dr_US	_fl_oz_US/8
Mass	usquarter	_qtr_US	25 * _lb
	ushundredweight	_cwt_US	4 * _qtr_US
	uston	_ton_US	20 * _cwt_US

## 4.10 International Currencies

International currency units based on exchange rates from 7.3.2019, 21:00 UTC.

Name	Symbol	Definition
AfghanAfghani	_AFN	$0.012 * \_EUR$
AfghanPul	_cAFN	$0.01 * \_AFN$
AlbanianLek	_ALL	$0.008 * \_EUR$
ArmenianDram	_AMD	$0.0018 * \_EUR$
ArmenianLuma	_cAMD	$0.01 * \_AMD$
AngolanKwanza	_AOA	$0.0028 * \_EUR$
AngolanCentimo	_cAOA	$0.01 * \_AOA$
ArgentinePeso	_ARS	$0.021 * \_EUR$
ArgentineCentavo	_cARS	$0.01 * \_ARS$
AustralianDollar	_AUD	$0.63 * \_EUR$
AustralianCent	_cAUD	$0.01 * \_AUD$
AzerbaijaniManat	_AZN	$0.63 * \_EUR$
AzerbaijaniQepik	_cAZN	$0.01 * \_AZN$
BosnianMark	_BAM	$0.51 * \_EUR$
BosnianFenings	_cBAM	$0.01 * \_BAM$
BangladeshiTaka	_BDT	$0.011 * \_EUR$
BangladeshiPoisha	_cBDT	$0.01 * \_BDT$
BurundianFranc	_BIF	$0.00049 * \_EUR$
BurundianCentime	_cBIF	$0.01 * \_BIF$
BolivianBoliviano	_BOB	$0.13 * \_EUR$
BolivianCentavo	_cBOB	$0.01 * \_BOB$
BrazilianReal	_BRL	$0.23 * \_EUR$
BrazilianCentavo	_cBRL	$0.01 * \_BRL$
BotswanaPula	_BWP	$0.083 * \_EUR$
BotswanaThebe	_cBWP	$0.01 * \_BWP$
BelarusianRuble	_BYN	$0.42 * \_EUR$
BelarusianKapiejka	_cBYN	$0.01 * \_BYN$



Name	Symbol	Definition
CanadianDollar	_CAD	0.66 * _EUR
CanadianCent	_cCAD	0.01 * _CAD
CongoleseFranc	_CDF	0.00055 * _EUR
CongoleseCentime	_cCDF	0.01 * _CDF
SwissFranc	_CHF	0.88 * _EUR
SwissRappen	_cCHF	0.01 * _CHF
ChileanPeso	_CLP	0.0013 * _EUR
ChileanCentavo	_cCLP	0.01 * _CLP
ChineseRenminbiYuan	_CNY	0.13 * _EUR
ChineseRenminbiFen	_cCNY	0.01 * _CNY
ColombianPeso	_COP	0.00028 * _EUR
ColombianCentavo	_cCOP	0.01 * _COP
CostaRicanColon	_CRC	0.0015 * _EUR
CostaRicanCentimos	_cCRC	0.01 * _CRC
CzechKoruna	_CZK	0.039 * _EUR
CzechHaler	_cCZK	0.01 * _CZK
DanishKrone	_DKK	0.13 * _EUR
DanishOre	_cDKK	0.01 * _DKK
DominicanPeso	_DOP	0.018 * _EUR
DominicanCentavo	_cDOP	0.01 * _DOP
AlgerianDinar	_DZD	0.0074 * _EUR
AlgerianSanteem	_cDZD	0.01 * _DZD
EgyptianPound	_EGP	0.051 * _EUR
EgyptianPiastre	_cEGP	0.01 * _EGP
EthiopianBirr	_ETB	0.031 * _EUR
EthiopianSantim	_cETB	0.01 * _ETB
FijianDollar	_FJD	0.42 * _EUR
FijianCent	_cFJD	0.01 * _FJD

Name	Symbol	Definition
PoundSterling	_GBP	1.16 * _EUR
PennySterling	_cGBP	0.01 * _GBP
GeorgianLari	_GEL	0.33 * _EUR
GeorgianTetri	_cGEL	0.01 * _GEL
GhanaianCedi	_GHS	0.16 * _EUR
GhanaianPesewa	_cGHS	0.01 * _GHS
GambianDalasi	_GMD	0.018 * _EUR
GambianButut	_cGMD	0.01 * _GMD
GuineanFranc	_GNF	9.6e-05 * _EUR
GuineanCentime	_cGNF	0.01 * _GNF
GuatemalanQuetzal	_GTQ	0.12 * _EUR
GuatemalanCentavo	_cGTQ	0.01 * _GTQ
GuyaneseDollar	_GYD	0.0043 * _EUR
GuyaneseCent	_cGYD	0.01 * _GYD
HongKongDollar	_HKD	0.11 * _EUR
HongKongCent	_cHKD	0.01 * _HKD
HonduranLempira	_HNL	0.036 * _EUR
HonduranCentavo	_cHNL	0.01 * _HNL
CroatianKuna	_HRK	0.13 * _EUR
CroatianLipa	_cHRK	0.01 * _HRK
HaitianGourde	_HTG	0.011 * _EUR
HaitianCentime	_cHTG	0.01 * _HTG
HungarianForint	_HUF	0.0032 * _EUR
HungarianFiller	_cHUF	0.01 * _HUF
IndonesianRupiah	_IDR	6.2e-05 * _EUR
IndonesianSen	_cIDR	0.01 * _IDR
IsraeliNewShekel	_ILS	0.25 * _EUR
IsraeliNewAgora	_cILS	0.01 * _ILS

Name	Symbol	Definition
IndianRupee	_INR	0.013 * _EUR
IndianPaissa	_cINR	0.01 * _INR
IraqiDinar	_IQD	0.00074 * _EUR
IraqiFils	_cIQD	0.001 * _IQD
IranianRial	_IRR	2.7e-05 * _EUR
IranianToman	_cIRR	10.0 * _IRR
IcelandicKrona	_ISK	0.0073 * _EUR
JamaicanDollar	_JMD	0.007 * _EUR
JamaicanCent	_cJMD	0.01 * _JMD
JapaneseYen	_JPY	0.008 * _EUR
KenyanShilling	_KES	0.0089 * _EUR
KenyanCent	_cKES	0.01 * _KES
KyrgyzstaniSom	_KGS	0.013 * _EUR
KyrgyzstaniTyin	_cKGS	0.01 * _KGS
CambodianRiel	_KHR	0.00022 * _EUR
NorthKoreanWon	_KPW	0.00099 * _EUR
NorthKoreanChon	_cKPW	0.01 * _KPW
SouthKoreanWon	_KRW	0.00078 * _EUR
SouthKoreanJeon	_cKRW	0.01 * _KRW
KuwaitiDinar	_KWD	2.93 * _EUR
KuwaitiFils	_cKWD	0.001 * _KWD
KazakhstaniTenge	_KZT	0.0023 * _EUR
KazakhstaniTiyn	_cKZT	0.01 * _KZT
LaoKip	_LAK	0.0001 * _EUR
LaoAtt	_cLAK	0.01 * _LAK
SriLankanRupee	_LKR	0.005 * _EUR
SriLankanCent	_cLKR	0.01 * _LKR
LiberianDollar	_LRD	0.0055 * _EUR
LiberianCent	_cLRD	0.01 * _LRD

Name	Symbol	Definition
LibyanDinar	_LYD	0.64 * _EUR
LibyanDirham	_cLYD	0.001 * _LYD
MoroccanDirham	_MAD	0.092 * _EUR
MoroccanSantim	_cMAD	0.01 * _MAD
MoldovanLeu	_MDL	0.052 * _EUR
MoldovanBan	_cMDL	0.01 * _MDL
MalagasyAriary	_MGA	0.00025 * _EUR
MalagasyIraimbilanja	_cMGA	0.2 * _MGA
MacedonianDenar	_MKD	0.016 * _EUR
MacedonianDeni	_cMKD	0.01 * _MKD
BurmeseKyat	_MMK	0.00059 * _EUR
BurmesePya	_cMMK	0.01 * _MMK
MongolianTogrog	_MNT	0.00034 * _EUR
MongolianMongol	_cMNT	0.01 * _MNT
MauritanianOuguiya	_MRU	0.025 * _EUR
MauritanianKhoudms	_cMRU	0.2 * _MRU
MauritianRupee	_MUR	0.025 * _EUR
MauritianCent	_cMUR	0.01 * _MUR
MaldivianRufiyaa	_MVR	0.058 * _EUR
MaldivianLaari	_cMVR	0.01 * _MVR
MalawianKwacha	_MWK	0.0012 * _EUR
MalawianTambala	_cMWK	0.01 * _MWK
MexicanPeso	_MXN	0.046 * _EUR
MexicanCentavo	_cMXN	0.01 * _MXN
MalaysianRinggit	_MYR	0.22 * _EUR
MalaysianSen	_cMYR	0.01 * _MYR
MozambicanMetical	_MZN	0.014 * _EUR
MozambicanCentavo	_cMZN	0.01 * _MZN

Name	Symbol	Definition
NigerianNaira	_NGN	0.0025 * _EUR
NigerianKobo	_cNGN	0.01 * _NGN
NicaraguanCordoba	_NIO	0.027 * _EUR
NicaraguanCentavo	_cNIO	0.01 * _NIO
NorwegianKrone	_NOK	0.1 * _EUR
NorwegianOre	_cNOK	0.01 * _NOK
NewZealandDollar	_NZD	0.61 * _EUR
NewZealandCent	_cNZD	0.01 * _NZD
PeruvianSol	_PEN	0.27 * _EUR
PeruvianCentimo	_cPEN	0.01 * _PEN
PapuaNewGuineanKina	_PGK	0.26 * _EUR
PapuaNewGuineanToea	_cPGK	0.01 * _PGK
PhilippinePeso	_PHP	0.017 * _EUR
PhilippineSentimo	_cPHP	0.01 * _PHP
PakistaniRupee	_PKR	0.0064 * _EUR
PakistaniPaissa	_cPKR	0.01 * _PKR
PolishZloty	_PLN	0.23 * _EUR
PolishGrosz	_cPLN	0.01 * _PLN
ParaguayanGuarani	_PYG	0.00015 * _EUR
ParaguayanCentimo	_cPYG	0.01 * _PYG
QatariRiyal	_QAR	0.24 * _EUR
QatariDirham	_cQAR	0.01 * _QAR
RomanianLeu	_RON	0.21 * _EUR
RomanianBan	_cRON	0.01 * _RON
SerbianDinar	_RSD	0.0085 * _EUR
SerbianPara	_cRSD	0.01 * _RSD
RussianRuble	_RUB	0.013 * _EUR
RussianKopeyka	_cRUB	0.01 * _RUB

Name	Symbol	Definition
RwandanFranc	_RWF	0.00098 * _EUR
RwandanCentime	_cRWF	0.01 * _RWF
SolomonIslandsDollar	_SBD	0.11 * _EUR
SolomonIslandsCent	_cSBD	0.01 * _SBD
SeychelloisRuppee	_SCR	0.065 * _EUR
SeychelloisCent	_cSCR	0.01 * _SCR
SudanesePound	_SDG	0.019 * _EUR
SudaneseQirsh	_cSDG	0.01 * _SDG
SwedishKrona	_SEK	0.094 * _EUR
SwedishOre	_cSEK	0.01 * _SEK
SingaporeDollar	_SGD	0.65 * _EUR
SingaporeCent	_cSGD	0.01 * _SGD
SierraLeoneanLeone	_SLL	0.0001 * _EUR
SierraLeoneanCent	_cSLL	0.01 * _SLL
SomalilandShilling	_SQS	0.00013 * _EUR
SomalilandCent	_cSQS	0.01 * _SQS
SomaliShilling	_SOS	0.0015 * _EUR
SomaliSenti	_cSOS	0.01 * _SOS
SurinameseDollar	_SRD	0.12 * _EUR
SurinameseCent	_cSRD	0.01 * _SRD
SyrianPound	_SYP	0.0017 * _EUR
SyrianPiastre	_cSYP	0.01 * _SYP
ThaiBaht	_THB	0.028 * _EUR
ThaiSatang	_cTHB	0.01 * _THB
TajikistaniSamani	_TJS	0.094 * _EUR
TajikistaniDiram	_cTJS	0.01 * _TJS
Tonganpaanga	_TOP	0.397 * _EUR
TonganSeniti	_cTOP	0.01 * _TOP

Name	Symbol	Definition
TurkishLira	_TRY	0.16 * _EUR
TurkishKurus	_cTRY	0.01 * _TRY
TrinidadAndTobagoDollar	_TTD	0.13 * _EUR
TrinidadAndTobagoCent	_cTTD	0.01 * _TTD
NewTaiwanDollar	_TWD	0.029 * _EUR
NewTaiwanCent	_cTWD	0.01 * _TWD
TanzanianShilling	_TZS	0.00038 * _EUR
TanzanianSenti	_cTZS	0.01 * _TZS
UkrainianHryvnia	_UAH	0.00038 * _EUR
UkrainianKopiyka	_cUAH	0.01 * _UAH
UgandanShilling	_UGX	0.00024 * _EUR
UgandanCent	_cUGX	0.01 * _UGX
USDollar	_USD	0.89 * _EUR
USCent	_cUSD	0.01 * _USD
UruguayanPeso	_UYU	0.027 * _EUR
UruguayanCentesimo	_cUYU	0.01 * _UYU
UzbekistaniSom	_UZS	0.00011 * _EUR
UzbekistaniTiyin	_cUZS	0.01 * _UZS
VenezuelanBolivarSoberano	_VES	0.0003 * _EUR
VenezuelanCentimoSoberano	_cVES	0.01 * _VES
VietnameseDong	_VND	3.8e-05 * _EUR
VietnameseXu	_cVND	0.01 * _VND
SamoanTala	_WST	0.34 * _EUR
SamoanSene	_cWST	0.01 * _WST
YemeniRial	_YER	0.0036 * _EUR
YemeniDinar	_cYER	0.01 * _YER
SouthAfricanRand	_ZAR	0.062 * _EUR
SouthAfricanCent	_cZAR	0.01 * _ZAR
ZambianKwacha	_ZMW	0.074 * _EUR
ZambianNgwee	_cZMW	0.01 * _ZMW

#### 4.10.1 Pegged International Currencies

International currency which are pegged to other currencies.

Name	Symbol	Definition
UnitedArabEmiratesDirham	_AED	$(1/3.6725) * \text{\_USD}$
UnitedArabEmiratesFils	_cAED	$0.01 * \text{\_AED}$
NetherlandsAntilleanGuilder	_ANG	$(1/1.79) * \text{\_USD}$
NetherlandsAntilleanCent	_cANG	$0.01 * \text{\_ANG}$
ArubanFlorin	_AWG	$(1/1.79) * \text{\_USD}$
ArubanCent	_cAWG	$0.01 * \text{\_AWG}$
BarbadianDollar	_BBD	$0.5 * \text{\_USD}$
BarbadianCent	_cBBD	$0.01 * \text{\_BBD}$
BulgarianLev	_BGN	$0.51129 * \text{\_EUR}$
BulgarianStotinka	_cBGN	$0.01 * \text{\_BGN}$
BahrainiDinar	_BHD	$(1/0.376) * \text{\_USD}$
BahrainiFils	_cBHD	$0.001 * \text{\_BHD}$
BermudianDollar	_BMD	$1 * \text{\_USD}$
BermudianCent	_cBMD	$0.01 * \text{\_BMD}$
BruneiDollar	_BND	$1 * \text{\_SGD}$
BruneiSen	_cBND	$0.01 * \text{\_BND}$
BahamianDollar	_BSD	$1 * \text{\_USD}$
BahamianCent	_cBSD	$0.01 * \text{\_BSD}$
BhutaneseNgultrum	_BTN	$1 * \text{\_INR}$
BhutaneseChhertum	_cBTN	$0.01 * \text{\_BTN}$
BelizeDollar	_BZD	$0.5 * \text{\_USD}$
BelizeCent	_cBZD	$0.01 * \text{\_BZD}$
CubanoConvertiblePeso	_CUC	$1 * \text{\_USD}$
CubanoConvertibleCentavo	_cCUC	$0.01 * \text{\_CUC}$
CubanPeso	_CUP	$(1/24) * \text{\_CUC}$
CubanCentavo	_cCUP	$0.01 * \text{\_CUP}$



Name	Symbol	Definition
CapeVerdeanEscudo	_CVE	$(1/110.265) * \text{\_EUR}$
CapeVerdeanCentavo	_cCVE	$0.01 * \text{\_CVE}$
DjiboutianFranc	_DJF	$(1/177.721) * \text{\_USD}$
DjiboutianCentime	_cDJF	$0.01 * \text{\_DJF}$
EritreanNakfa	_ERN	$(1/15) * \text{\_USD}$
EritreanCent	_cERN	$0.01 * \text{\_ERN}$
FalklandIslandsPound	_FKP	$1 * \text{\_GBP}$
FalklandIslandsPenny	_cFKP	$0.01 * \text{\_FKP}$
GuernseyPound	_GGP	$1 * \text{\_GBP}$
GuernseyPenny	_cGGP	$0.01 * \text{\_GGP}$
GibraltarPound	_GIP	$1 * \text{\_GBP}$
GibraltarPenny	_cGIP	$0.01 * \text{\_GIP}$
ManxPound	_IMP	$1 * \text{\_GBP}$
ManxPenny	_cIMP	$0.01 * \text{\_IMP}$
JerseyPound	_JEP	$1 * \text{\_GBP}$
JerseyPenny	_cJEP	$0.01 * \text{\_JEP}$
JordanianDinar	_JOD	$(1/0.708) * \text{\_USD}$
JordanianFils	_cJOD	$0.001 * \text{\_JOD}$
KiribatiDollar	_KID	$1 * \text{\_AUD}$
KiribatiCent	_cKID	$0.01 * \text{\_KID}$
Comorianfranc	_KMF	$(1/491.96775) * \text{\_EUR}$
ComorianCentime	_cKMF	$0.01 * \text{\_KMF}$
CaymanIslandsDollar	_KYD	$1.2 * \text{\_USD}$
CaymanIslandsCent	_cKYD	$0.01 * \text{\_KYD}$
LebanesePound	_LBP	$(1/1507.5) * \text{\_USD}$
LebaneseQeresh	_cLBP	$0.01 * \text{\_LBP}$
MacanesePataca	_MOP	$(1/1.03) * \text{\_HKD}$
MacaneseAvo	_cMOP	$0.01 * \text{\_MOP}$

Name	Symbol	Definition
NamibianDollar	_NAD	1 * _ZAR
NamibianCent	_cNAD	0.01 * _NAD
NepaleseRupee	_NPR	(1/1.6) * _INR
NepalesePaissa	_cNPR	0.01 * _NPR
OmaniRial	_OMR	(1/2.6008) * _USD
OmaniBaisa	_cOMR	0.001 * _OMR
PanamanianBalboa	_PAB	1 * _USD
PanamanianCentesimo	_cPAB	0.01 * _PAB
TransnistrianRuble	_PRB	(1/16.1) * _USD
TransnistrianKopeck	_cPRB	0.01 * _PRB
SaudiRiyal	_SAR	(1/3.75) * _USD
SaudiHalalah	_cSAR	0.01 * _SAR
SaintHelenaPound	_SHP	1 * _GBP
SaintHelenaPenny	_cSHP	0.01 * _SHP
SouthSudanesePound	_SSP	1 * _SDG
SouthSudanesePiaster	_cSSP	0.01 * _SSP
SaoTomeAndPrincipeDobra	_STN	(1/24.5) * _EUR
SaoTomeAndPrincipeCentimo	_cSTN	0.01 * _STN
SwaziLilangeni	_SZL	1 * _ZAR
SwaziCent	_cSZL	0.01 * _SZL
TurkmenistanManat	_TMT	(1/3.5) * _USD
TurkmenistanTenge	_cTMT	0.01 * _TMT
TuvaluanDollar	_TVD	1 * _AUD
TuvaluanCent	_cTVD	0.01 * _TVD
CentralAfricanCFAFranc	_XAF	(1/655.957) * _EUR
CentralAfricanCFACentime	_cXAF	0.01 * _XAF
EasternCaribbeanDollar	_XCD	(1/2.7) * _USD
EasternCaribbeanCent	_cXCD	0.01 * _XCD

Name	Symbol	Definition
WestAfricanCFAFranc	_XOF	$(1/655.957) * \text{\_USD}$
WestAfricanCFACentime	_cXOF	$0.01 * \text{\_XOF}$
CFPFranc	_XPF	$(1000/8.38) * \text{\_EUR}$
CFPCentime	_cXPF	$0.01 * \text{\_XPF}$
ZimbabweanBonds	_ZWL	$1 * \text{\_USD}$
ZimbabweanCent	_cZWL	$0.01 * \text{\_ZWL}$

## 5 Lua Documentation

In this section, the following shortcuts will be used.

```
local D = physical.Dimension
local U = physical.Unit
local N = physical.Number
local Q = physical.Quantity
```

The term **number** refers to a lua integer or a lua float number. By **string** a lua string is meant and by **bool** a lua boolean.

### 5.1 physical.Quantity

The quantity class is the main part of the library. Each physical Quantity and all units are represented by an instance of this class.

#### **Q.new(q=nil)**

Copy Constuctor

**q** : Q, number, object, nil

returns : Q

As an argument it takes Q, **number**, **object** or **nil**. If an instance of Q is given, a copy is made and returned. If a **number** or an instance **object** of another class is given, the function creates a dimensionless quantity with the **number** or the instance as a value. In the case **nil** is given, a dimensionless quantity with value 1 is returned.

```
print( Q() )
1

print( Q(42) )
42

print( Q(73*_m) )
73 * _m
```

#### **Q.defineBase(symbol,name,dimension)**

This function is used to declare base quantities from which all other quantities are derived from.

**symbol** : string

The symbol of the base quantity.

**name** : string  
The name of the base quantity.

**dimension** : D  
An instance of the D class, which represents the dimension of the quantity.

returns : Q  
The created Q instance.

The function creates a global variable of the created base quantity. The name consist of an underscore concatenated with the **symbol** argument, i.e. the symbol **m** becomes the global variable **\_m**.

The **name** is used for example in the siunitx conversion function, e.g **meter** will be converted to **\meter**.

Each quantity has a dimension associated with it. The argument **dimension** allows any dimension to be associated to base quantities.

```
Q.defineBase("m", "meter", L)
Q.defineBase("kg", "kilogram", M)
```

## **Q.define(symbol, name, q)**

Creates a new derived quantity from an expression of other quantities. Affine quantities like the absolute temperature in celsius are not supported.

**symbol** : string  
Symbol of the base quantity

**name** : string, nil  
The Name of the derived quantity.

**q** : physical.Quantity  
The definition of the derived quantity.

returns : Q  
The created quantity.

The function creates a global variable of the created base quantity. The name consist of an underscore concatenated with the **symbol** argument, i.e. the symbol **N** becomes the global variable **\_N**.

The **name** is used for example in the siunitx conversion function, e.g **newton** will be converted to **\newton**.

```
Q.define("L", "liter", _dm^3)
Q.define("Pa", "pascal", _N/_m^2)
Q.define("C", "coulomb", _A*_s)

Q.define("degC", "celsius", _K)
```

### **Q.definePrefix(symbol,name,factor)**

Defines a new prefix.

**symbol** : string  
Symbol of the base quantity  
**name** : string  
Name of the base quantity  
**factor** : number  
The factor which corresponds to the prefix

```
Q.definePrefix("c", "centi", 1e-2)
Q.definePrefix("a", "atto", 1e-18)
```

### **Q.addPrefix(prefixes, units)**

Create several units with prefixes from a given unit.

**prefixes** : string  
A list of unit symbols.  
**units** : Q  
A list of quantities.

```
Q.addPrefix({"n","u","m","k","M","G"},{_m,_s,_A})
```

### **Q.isclose(self,q,r)**

Checks if this quantity is close to another one. The argument **r** is the maximum relative deviation. The function returns **true** if the following condition is fulfilled

$$\frac{abs(self - q)}{min(self,q)} \leq r \quad .$$

**self** : Q, N, number  
**q** : Q, N, number  
**r** : number  
maximum relative deviation of **self** and **q**  
returns : bool  
true if **q** is close to **self**, otherwise false

```
s_1 = 1.9 * _m
s_2 = 2.0 * _m
print( s_1:isclose(s_2,0.1) )
true
print( s_1:isclose(s_2,0.01) )
false
```

### **Q.to(self,q=nil)**

Converts the quantity **self** to the unit of the quantity **q**. If no **q** is given, the quantity **self** is converted to base units.

```
self : Q
q : Q, nil
```

```
s = 1.9 * _km
print( s:to(_m) )
1900.0 * _m

T = 10 * _degC
print( T:to(_K) )
10.0 * _K

print( T:to() )
10 * _K
```

### **Q.tosiunitx(self,param,mode=Q.siunitx\_SI)**

Converts the quantity into a siunitx string.

```
self : Q
param : string
mode : number
```

If **mode** is equal **Q.SIUNITX\_SI**, which is the default, the quantity is converted to an `\SI{...}{...}` macro. If **mode** is **Q.SIUNITX\_num**, the quantity is converted to `\num{...}` and if it is **Q.SIUNITX\_si** the macro `\si{...}` is printed.

```
s = 1.9 * _km

print( s:tosiunitx() )
\SI{1.9}{\kilo\meter}

print( s:tosiunitx(nil,Q.SIUNITX_num) )
\num{1.9}

print( s:tosiunitx(nil,Q.SIUNITX_si) )
\si{\kilo\meter}
```

### **Q.min(q1, q2, ...)**

Returns the smallest quantity of the given ones. The function returns q1 if the Quantities are equal.

q1 : Q, N, number

q2 : Q, N, number

...

qN : Q, N, number

returns : Q

the smallest quantity of q1, ... , qN

```
s_1 = 15 * _m
s_2 = 5 * _m
print(s_1:min(s_2))
5 * _m
```

### **Q.max(q1, q2, ...)**

Returns the biggest quantity of several given ones. The function returns q1 if the Quantities are equal.

q1 : Q, N, number

q2 : Q, N, number

...

qN : Q, N, number

returns : Q

the biggest quantity of q1, ... , qN

```
s_1 = 15 * _m
s_2 = 5 * _m
print(s_1:max(s_2))
15 * _m
```

### **Q.abs(q)**

Returns the absolute value of the given quantity q.

q : Q, N, number

returns : Q

the absolute value of q



```

U = -5 * _V
print(U)
-5 * _V
print(U:abs())
5 * _V

```

### **Q.sqrt(q)**

Returns the square root of the given quantity.

**q** : Q, N, number  
dimensionless argument

returns : Q  
the square root of **q**

```

A = 25 * _m^2
s = A:sqrt()
print(s)
5.0 * _m

```

### **Q.log(q, base=nil)**

Returns the logarithm of a given quantity to the given base. If no base is given, the natural logarithm is returned.

**q** : Q, N, number  
dimensionless argument

**base** : Q, N, number, nil  
dimensionless argument

returns : Q  
logarithm of **q** to the **base**

```

I = 1 * _W/_m^2
I_0 = 1e-12 * _W/_m^2
print(10 * (I/I_0):log(10) * _dB )
120.0 * _dB

```

### **Q.exp(q)**

Returns the value of the natural exponential function of the given quantity.

**q** : Q, N, number  
dimensionless argument

returns : Q  
natural exponential of q

```
x = 2 * _1
print( x:exp() )
7.3890560989307
```

### **Q.sin(q)**

Returns the value of the sinus function of the given quantity.

q : Q, N, number  
dimensionless argument

returns : Q  
sine of q

```
alpha = 30 * _deg
print( alpha:sin() )
0.5
```

### **Q.cos(q)**

Returns the value of the cosinus function of the given quantity. The quantity has to be dimensionless.

q : Q, N, number  
dimensionless argument

returns : Q  
cosine of q

```
alpha = 60 * _deg
print( alpha:cos() )
0.5
```

### **Q.tan(q)**

Returns the value of the tangent function of the given quantity. The quantity has to be dimensionless.

q : Q, N, number  
dimensionless argument

returns : Q  
tangent of q

```
alpha = 45 * _deg
print( alpha:tan() )
1.0
```

### **Q.asin(q)**

Returns the value of the arcus sinus function of the given quantity.  
The quantity has to be dimensionless.

**q** : Q, N, number  
dimensionless argument  
**returns** : Q  
inverse sine of **q**

```
x = 0.5 * _1
print( x:asin():to(_deg) )
30.0 * _deg
```

### **Q.acos(q)**

Returns the value of the arcus cosinus function of the given quantity.  
The quantity has to be dimensionless.

**q** : Q, N, number  
dimensionless argument  
**returns** : Q  
inverse cosine of **q**

```
x = 0.5 * _1
print( x:acos():to(_deg) )
60.0 * _deg
```

### **Q.atan(q)**

Returns the value of the arcus tangent function of the given quantity.  
The quantity has to be dimensionless.

**q** : Q, N, number  
dimensionless argument  
**returns** : Q  
inverse tangent of **q**

```
x = 1 * _1
print( x:atan():to(_deg) )
45.0 * _deg
```

### **Q.sinh(q)**

Returns the value of the hyperbolic sine function of the given quantity. The quantity has to be dimensionless. Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\sinh(x) = 0.5 \cdot e^x - 0.5/e^x \quad .$$

**q** : Q, N, number  
dimensionless argument  
returns : Q  
hyperbolic sine of q

```
x = 1 * _1
print( x:sinh() )
1.1752011936438
```

### **Q.cosh(q)**

Returns the value of the hyperbolic cosine function of the given quantity. The quantity has to be dimensionless. Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\cosh(x) = 0.5 \cdot e^x + 0.5/e^x \quad .$$

**q** : Q, N, number  
dimensionless argument  
returns : Q  
hyperbolic cosine of q

```
x = 1 * _1
print( x:cosh() )
1.5430806348152
```

### **Q.tanh(q)**

Returns the value of the hyperbolic tangent function of the given quantity. The quantity has to be dimensionless. Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad .$$

**q** : Q, N, number  
dimensionless argument  
returns : Q  
hyperbolic tangent of q

```
x = 1 * _1
print( x:tanh() )
0.76159415595576
```

### Q.asinh(q)

Returns the value of the inverse hyperbolic sine function of the given quantity. The quantity has to be dimensionless. Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\operatorname{asinh}(x) = \ln \left( x + \sqrt{x^2 + 1} \right) \quad .$$

q : Q, N, number  
dimensionless argument  
returns : Q  
inverse hyperbolic sine of q

```
x = 1 * _1
print( x:asinh() )
0.88137358701954
```

### Q.acosh(q)

Returns the value of the inverse hyperbolic cosine function of the given quantity. The quantity has to be dimensionless. Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\operatorname{acosh}(x) = \ln \left( x + \sqrt{x^2 - 1} \right) \quad , x > 1 \quad .$$

q : Q, N, number  
dimensionless argument bigger or equal to one  
returns : Q  
inverse hyperbolic cosine of q

```
x = 2 * _1
print( x:acosh() )
1.3169578969248
```

### Q.atanh(q)

Returns the value of the inverse hyperbolic tangent function of the given quantity. The quantity has to be dimensionless. Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\operatorname{atanh}(x) = \ln \left( \frac{1+x}{1-x} \right) \quad , -1 < x < 1 \quad .$$

**q** : Q, N, number  
dimensionless argument with magnitude smaller than one  
**returns** : Q  
inverse hyperbolic tangent of **q**

```
x = 0.5 * _1
print( x:atanh() )
0.54930614433405
```

## 5.2 physical.Dimension

All physical quantities do have a physical dimension. For example the quantity *Area* has the dimension  $L^2$  (length to the power of two). In the SI-System there are seven base dimensions, from which all other dimensions are derived. Each dimension is represented by an  $n$ -tuple, where  $n$  is the number of base dimensions. Each physical quantity has an associated dimension object. It is used to check equality and if addition or subtraction is allowed.

### D.new(d)

Constructor of the `Dimension` class.

`d` : `Dimension` or `string`, `nil`  
The name or symbol of the dimension.  
returns : `D`  
The created `D` instance

If `d` is a string, a copy of the perviously defined dimension is made.  
If `d` is a dimension, a copy of it is made. If no argument ist given, a dimension *zero* is created.

#### Example

```
V_1 = D("Velocity")
L = D("L")
V_2 = D(L/T)
```

### D.defineBase(symbol, name)

Defines a base dimension.

`symbol` : `string`  
  
`name` : `string`  
  
returns : `D`  
The created `D` instance

#### Example

```
V_1 = D("Velocity")
L = D("L")
V_2 = D(L/T)
```

### 5.3 physical.Unit

The task of this class is keeping track of the unit term. The unit term is a fraction of units. The units in the enumerator and denominator can have an exponent.

#### **Unit.new(u=nil)**

Copy Constructor. It copies a given unit object. If nothing is given, an empty unit is created.

**u : Unit**  
The unit object which will be copied.  
**returns : Unit**  
The created **Unit** object

#### **Unit.new(symbol, name, prefixsymbol=nil, prefixname=nil)**

Constructor. A new **Unit** object with symbol is created. The prefixsymbol and prefixname are optional.

**symbol : String**  
The symbol of the unit.  
**name : String**  
The name of the unit.  
**prefixsymbol : String**  
The optional symbol of the prefix.  
**prefixname : String**  
The optional name of the prefix.  
**returns : Unit**  
The created **Unit** object

#### **Unit.tosiunitx(self)**

The unit term will be compiled into a string, which the LaTeX package siunitx can understand.

**returns : String**  
The siunitx representation of the unit term.



## 5.4 physical.Number

It does arithmetics with gaussian error propagation. A number instance has a mean value called **x** and an uncertainty value called **dx**.

### **N.new(n=nil)**

This is the copy Constructor. It copies a given number object. If **n** is **nil**, an instance representing number zero with uncertainty zero is created.

**n** : **Number**  
The number object to be copied.  
**returns** : **Number**  
The created **Number** instance.

```
n = N(56,0.012)
m = N(n)
print(m)
(56.000 +/- 0.012)
```

### **N.new(x, dx=nil)**

This constructor, creates a new instance of **N** with mean value **x** and uncertainty **dx**. If **dx** is not given, the uncertainty is zero.

**x** : **number**  
mean value  
**dx** : **number, nil**  
uncertainty value  
**returns** : **N**  
The created **N** instance.

```
n = N(56,0.012)
print(n)
(56.000 +/- 0.012)
```

### **N.new(str)**

This constructor creates a new instance of **N** from a string. It can parse strings of the form "3.4", "3.4e-3", "5.4e-3 +/- 2.4e-6" and "5.45(7)e-23".

**str** : **string**  
**returns** : **N**

```

n_1 = N("12.3e-3")
print(n_1)
(0.01230 +/- 0.00005)

n_2 = N("12 +/- 0.1")
print(n_2)
(12.00 +/- 0.10)

n_3 = N("12.0(1)")
print(n_3)
(12.00 +/- 0.10)

n_4 = N("15.0(12)")
print(n_4)
(15.0 +/- 1.2)

```

### **N.mean(n)**

Returns the mean value of **n**.

#### **Parameters / Return**

returns : number

```

n = N(1.25,0.0023)
print( n:mean() )
1.25

```

### **N.uncertainty(n)**

Returns the uncertainty value of **n**.

**n** : N

returns : number

```

n = N(1.25,0.0023)
print( n:uncertainty() )
0.0023

```

### **N.abs(n)**

Returns the absolute value of **n**.

**n** : N

returns : N

The uncertainty  $\Delta y$  is calculated by the following expression

$$\Delta y = \Delta x \quad .$$

```
n = N(-10,1)
print( n:abs() )
(10.0 +/- 1.0)
```

### **N.sqrt(n)**

Returns the square root of **n**.

**n** : N  
returns : N

The uncertainty  $\Delta y$  is calculated by the following expression

$$\Delta y = \frac{1}{2\sqrt{x}} \cdot \Delta x \quad .$$

```
n = N(25,1)
print( n:sqrt() )
(5.00 +/- 0.10)
```

### **N.log(n,base=nil)**

Returns the logarithm of a given number **n** to the given base **base**. If no base is given, the natural logarithm of **n** is returned.

**n** : N  
**base** : number, nil  
returns : N

The uncertainty  $\Delta y$  is calculated by the following expression

$$\Delta y = \frac{1}{|x \cdot \log(b)|} \cdot \Delta x \quad .$$

```
n = N(25,1)
print( n:log() )
(3.22 +/- 0.04)
```

### **N.exp(n)**

Returns the value of the natural exponential function of the given number.

**q** : N  
returns : N

The uncertainty  $\Delta y$  is calculated by the following expression

$$\Delta y = e^x \cdot \Delta x \quad .$$

```
n = N(25,1)
print( n:sqrt() )
(5.00 +/- 0.10)
```

### **N.sin(n)**

Returns the value of the sine function of the given number.

**n** : N  
returns : N

The uncertainty  $\Delta y$  is calculated by the following expression

$$\Delta y = | \cos(x) | \cdot \Delta x \quad .$$

```
n = N(3,0.1)
print( n:sin() )
(0.14 +/- 0.10)
```

### **N.cos(n)**

Returns the value of the cosine function of the given number.

**n** : N  
returns : N

The uncertainty  $\Delta y$  is calculated by the following expression

$$\Delta y = | \sin(x) | \cdot \Delta x \quad .$$

```
n = N(0.5,0.01)
print( n:cos() )
(0.878 +/- 0.005)
```

### **N.tan(n)**

Returns the value of the tangent function of the given number.

**n** : N

returns : N

The uncertainty  $\Delta y$  is calculated by the following expression

$$\Delta y = \left| \frac{1}{\cos^2(x)} \right| \cdot \Delta x \quad .$$

```
n = N(1.5,0.01)
print( n:tan() )
(14.1 +/- 2.0)
```

### **N.asin(n)**

Returns the value of the inverse sine function of the given number.

**n** : N

returns : N

The uncertainty  $\Delta y$  is calculated by the following expression

$$\Delta y = \frac{1}{\sqrt{1-x^2}} \cdot \Delta x \quad .$$

```
n = N(0.99,0.1)
print( n:asin() )
(1.4 +/- 0.7)
```

### **N.acos(n)**

Returns the value of the inverse cosine function of the given number.

**n** : N

returns : N

The uncertainty  $\Delta y$  is calculated by the following expression

$$\Delta y = \frac{1}{\sqrt{1-x^2}} \cdot \Delta x \quad .$$

```
n = N(0.99,0.1)
print( n:acos() )
(0.1 +/- 0.7)
```

### **N.atan(n)**

Returns the value of the inverse tangent function of the given number.

**n** : N

returns : N

The uncertainty  $\Delta y$  is calculated by the following expression

$$\Delta y = \frac{1}{\sqrt{1+x^2}} \cdot \Delta x \quad .$$

```
n = N(1,0.1)
print( n:atan() )
(0.79 +/- 0.05)
```

### **N.sinh(q)**

Returns the value of the hyperbolic sine function of the given number.

**n** : N

returns : N

Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\sinh(x) = 0.5 \cdot e^x - 0.5/e^x \quad .$$

The uncertainty  $\Delta y$  is calculated by the following expression

$$\Delta y = (0.5 \cdot e^x + 0.5/e^x) \cdot \Delta x \quad .$$

```
n = N(1,0.1)
print( n:sinh() )
(1.18 +/- 0.15)
```

### **N.cosh(q)**

Returns the value of the hyperbolic cosine function of the given number.

**n** : N

returns : N

Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\cosh(x) = 0.5 \cdot e^x + 0.5/e^x \quad .$$

The uncertainty  $\Delta y$  is calculated by the following expression

$$\Delta y = (0.5 \cdot e^x - 0.5/e^x) \cdot \Delta x \quad .$$

```
n = N(1,0.1)
print( n:cosh() )
(1.54 +/- 0.12)
```

### **N.tanh(q)**

Returns the value of the hyperbolic tangent function of the given number.

**n** : N

returns : N

Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad .$$

The uncertainty  $\Delta y$  is calculated by the following expression

$$\Delta y = \frac{1}{(0.5 \cdot e^x + 0.5/e^x)^2} \cdot \Delta x \quad .$$

```
n = N(1,0.1)
print( n:tanh() )
(0.76 +/- 0.04)
```

### **Q.asinh(q)**

Returns the value of the inverse hyperbolic sine function of the given number.

**n** : N

returns : N

Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\operatorname{asinh}(x) = \ln \left( x + \sqrt{x^2 + 1} \right) \quad .$$

The uncertainty  $\Delta y$  is calculated by the following expression

$$\Delta y = \frac{1}{\sqrt{x^2 + 1}} \cdot \Delta x \quad .$$

```
n = N(1,0.1)
print( n:asinh() )
(0.88 +/- 0.07)
```

### Q.acosh(q)

Returns the value of the inverse hyperbolic cosine function of the given number.

**n** : N  
returns : N

Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\operatorname{acosh}(x) = \ln \left( x + \sqrt{x^2 - 1} \right) \quad , x > 1 \quad .$$

The uncertainty  $\Delta y$  is calculated by the following expression

$$\Delta y = \frac{1}{\sqrt{x^2 - 1}} \cdot \Delta x \quad .$$

```
n = N(1,0.1)
print( n:acosh() )
(0 +/- inf)
```

### Q.atanh(q)

Returns the value of the inverse hyperbolic tangent function of the given number.

**n** : N  
returns : N



Since Lua doesn't implement the hyperbolic functions, the following formula is used

$$\operatorname{atanh}(x) = \ln \left( \frac{1+x}{1-x} \right) \quad , -1 < x < 1 \quad .$$

The uncertainty  $\Delta y$  is calculated by the following expression

$$\Delta y = \frac{1}{|x^2 - 1|} \cdot \Delta x \quad .$$

```
n = N(1,0.1)
print( n:atanh() )
(inf +/- inf)
```

## 6 Change History

V1.0.3 (2020/09/09) **Minor release**

Changed foldername `physical` to `src`. Changed Classvariables `Q.siunitx_SI`, `Q.siunitx_num` and `Q.siunitx_si` to uppercase `Q.SIUNITX_SI`, `Q.SIUNITX_num` and `Q.SIUNITX_si`.

V1.0.2 (2020/09/07) **Minor release**

Path issues resolved. Docs corrected.

V1.0.1 (2020/09/05) **Minor release**

Files renamed.

V1.0.1 (2020/09/03) **First official release.**

## References

- [1] Webpage <https://physics.nist.gov/cuu/index.html>, August 2019.
- [2] Bureau International des Poids et Mesures. The international system of units (si), 2006.
- [3] Bureau International des Poids et Mesures. Resolutions of the 26th cgpm, November 2018.
- [4] Prša et al. Nominal values for selected solar and planetary quantities: IAU 2015 resolution B3. The Astronomical Journal, 152:41, August 2016.

## Index of Units

acre `_ac`, 21  
amperehour `_Ah`, 20  
amperesecond `_As`, 20  
angstrom `_angstrom`, 19  
arcminute `_arcmin`, 18  
arcsecond `_arcsec`, 18  
are `_are`, 19  
astronomicalunit `_au`, 19  
atomicmassunit , 16  
avogadronumber , 16  
  
bar `_bar`, 19  
barn `_barn`, 19  
becquerel `_Bq`, 17  
bitpersecond `_bps`, 20  
bohrmagneton , 16  
boltzmannconstant , 16  
byte `_B`, 20  
  
cable `_cbl`, 21  
celsius `_degC`, 17  
chain `_ch`, 21  
coulomb `_C`, 17  
cup `_cup`, 21  
curie `_Ci`, 20  
  
day `_d`, 18  
decibel `_dB`, 20  
degree `_deg`, 18  
dram `_dr`, 22  
  
electronmagneticmoment , 16  
electronmass , 16  
electronvolt `_eV`, 20  
elementarycharge , 16  
eulersnumber , 16  
  
fahrenheit `_degF`, 22  
farad `_F`, 17  
fathom `_ftm`, 21  
fermi `_fermi`, 19  
finestructureconstant , 16  
firkin `_fir`, 22  
  
fluidram `_fl_dr`, 21  
fluidounce `_fl_oz`, 21  
foot `_ft`, 21  
fortnight `_ftn`, 22  
furlong `_fur`, 21  
  
gallon `_gal`, 21  
gill `_gi`, 21  
gradian `_gon`, 19  
grain `_gr`, 22  
gramoftnt `_g_TNT`, 19  
gravitationalconstant , 16  
gray `_Gy`, 17  
  
hand `_hh`, 21  
hectare `_hectare`, 18  
henry `_H`, 17  
hertz `_Hz`, 17  
horsepower `_hp`, 22  
hour `_h`, 18  
hundredweight `_cwt`, 22  
  
inch `_in`, 21  
intbritishthermalunit `_BTU_it`, 22  
internationalcalorie `_cal_IT`, 19  
  
joule `_J`, 17  
  
katal `_kat`, 17  
kilopond `_kp`, 19  
knot `_kn`, 21  
  
league `_lea`, 21  
lightsecond `_ls`, 19  
lightyear `_ly`, 19  
liter `_L`, 18  
longton `_ton`, 22  
lumen `_lm`, 17  
lux `_lx`, 17  
  
metrichorsepower `_PS`, 20  
metric tablespoon `_Tbsp`, 19  
metric teaspoon `_tsp`, 19  
mile `_mi`, 21  
millimeterofmercury `_mmHg`, 19

minute `_min`, 18  
 molargasconstant , 16  
  
 nauticalleague `_nlea`, 21  
 nauticalmile `_nmi`, 21  
 neutronmagneticmoment , 16  
 neutronmass , 16  
 newton `_N`, 17  
 nibble `_nibble`, 20  
 nomjoveqradius `_Re_J_nom`, 18  
 nomjovmassparameter `_GM_J_nom`,  
     18  
 nomjovpolradius `_Rp_J_nom`, 18  
 nomsoleffttemperature `_T_S_nom`, 18  
 nomsolirradiance `_S_S_nom`, 18  
 nomsol luminosity `_L_S_nom`, 18  
 nomsolmassparameter `_GM_S_nom`,  
     18  
 nomsolradius `_R_S_nom`, 18  
 nomterreqradius `_Re_E_nom`, 18  
 nomtermmassparameter `_GM_E_nom`,  
     18  
 nomterrpolarradius `_Rp_E_nom`, 18  
 nuclearmagneton , 16  
  
 ohm `_Ohm`, 17  
 ounce `_oz`, 22  
  
 parsec `_pc`, 19  
 partsperbillion `_ppb`, 20  
 partspermillion `_ppm`, 20  
 partsperquadrillion `_ppq`, 20  
 partspertrillion `_ppt`, 20  
 pascal `_Pa`, 17  
 pennyweight `_dwt`, 22  
 percent `_percent`, 20  
 permille `_permille`, 20  
 pi , 16  
 pica `_pica`, 21  
 pint `_pint`, 21  
 planckconstant , 16  
 point `_pt`, 21  
 poiseuille `_Pl`, 20  
 pound `_lb`, 22  
 poundal `_pdl`, 22  
 poundforce `_lbf`, 22  
  
 poundforcepersquareinch `_psi`, 22  
 protonmagneticmoment , 16  
 protonmass , 16  
  
 quart `_qt`, 21  
 quarter `_qtr`, 22  
  
 rad `_Rad`, 20  
 radian `_rad`, 17  
 reducedplanckconstant , 16  
 rem `_rem`, 20  
 rod `_rd`, 21  
 rydbergconstant , 16  
  
 sennight `_sen`, 22  
 siemens `_S`, 17  
 sievert `_Sv`, 17  
 slug `_slug`, 22  
 spat `_sp`, 19  
 speedoflight , 16  
 standardatmosphere `_atm`, 19  
 standardgravity , 16  
 stefanboltzmannconstant , 16  
 steradian `_sr`, 17  
 stone `_st`, 22  
 svedberg `_svedberg`, 19  
  
 technicalatmosphere `_at`, 19  
 tesla `_T`, 17  
 thchembritishthermalunit `_BTU`, 22  
 thermochemicalcalorie `_cal`, 19  
 thou `_th`, 21  
 tonne `_t`, 18  
 tonoftnt `_t_TNT`, 19  
 torr `_Torr`, 19  
 troyounce `_oz_t`, 22  
 troypound `_lb_t`, 22  
 turn `_tr`, 19  
  
 uscup `_cup_US`, 23  
 usfluidram `_fl_dr_US`, 23  
 usfluidounce `_fl_oz_US`, 23  
 usgallon `_gal_US`, 23  
 usgill `_gi_US`, 23  
 ushundredweight `_cwt_US`, 23  
 uspint `_pint_US`, 23  
 usquart `_qt_US`, 23

usquarter \_qtr\_US, 23  
 ussurveyacre \_ac\_US, 23  
 ussurveycable \_cbl\_US, 23  
 ussurveychain \_ch\_US, 23  
 ussurveyfathom \_ftm\_US, 23  
 ussurveyfoot \_ft\_US, 23  
 ussurveyfurlong \_fur\_US, 23  
 ussurveyhand \_hh\_US, 23  
 ussurveyinch \_in\_US, 23  
 ussurveyleague \_lea\_US, 23  
 ussurveylink \_li\_US, 23  
 ussurveymile \_mi\_US, 23  
 ussurveyrod \_rd\_US, 23  
 ussurveyyard \_yd\_US, 23  
 ustablespoon \_Tbsp\_US, 23

usteaspoon \_tsp\_US, 23  
 uston \_ton\_US, 23  
  
 vacuumpermeability , 16  
 vacuumpermittivity , 16  
 volt \_V, 17  
 voltampere \_VA, 20  
  
 watt \_W, 17  
 watthour \_Wh, 20  
 wattsecond \_Ws, 20  
 weber \_Wb, 17  
 week \_wk, 19  
  
 yard \_yd, 21  
 year \_a, 19

## Index of Currencies

AfghanAfghani \_AFN, 24  
AfghanPul \_cAFN, 24  
AlbanianLek \_ALL, 24  
AlgerianDinar \_DZD, 25  
AlgerianSanteem \_cDZD, 25  
AngolanCentimo \_cAOA, 24  
AngolanKwanza \_AOA, 24  
ArgentineCentavo \_cARS, 24  
ArgentinePeso \_ARS, 24  
ArmenianDram \_AMD, 24  
ArmenianLuma \_cAMD, 24  
ArubanCent \_cAWG, 32  
ArubanFlorin \_AWG, 32  
AustralianCent \_cAUD, 24  
AustralianDollar \_AUD, 24  
AzerbaijaniManat \_AZN, 24  
AzerbaijaniQepik \_cAZN, 24

BahamianCent \_cBSD, 32  
BahamianDollar \_BSD, 32  
BahrainiDinar \_BHD, 32  
BahrainiFils \_cBHD, 32  
BangladeshiPoisha \_cBDT, 24  
BangladeshiTaka \_BDT, 24  
BarbadianCent \_cBBD, 32  
BarbadianDollar \_BBD, 32  
BelarusianKapiejka \_cBYN, 24  
BelarusianRuble \_BYN, 24  
BelizeCent \_cBZD, 32  
BelizeDollar \_BZD, 32  
BermudianCent \_cBMD, 32  
BermudianDollar \_BMD, 32  
BhutaneseChhertum \_cBTN, 32  
BhutaneseNgultrum \_BTN, 32  
BolivianBoliviano \_BOB, 24  
BolivianCentavo \_cBOB, 24  
BosnianFenings \_cBAM, 24  
BosnianMark \_BAM, 24  
BotswanaPula \_BWP, 24  
BotswanaThebe \_cBWP, 24  
BrazilianCentavo \_cBRL, 24  
BrazilianReal \_BRL, 24

BruneiDollar \_BND, 32  
BruneiSen \_cBND, 32  
BulgarianLev \_BGN, 32  
BulgarianStotinka \_cBGN, 32  
BurmeseKyat \_MMK, 28  
BurmesePya \_cMMK, 28  
BurundianCentime \_cBIF, 24  
BurundianFranc \_BIF, 24

CambodianRiel \_KHR, 27  
CanadianCent \_cCAD, 25  
CanadianDollar \_CAD, 25  
CapeVerdeanCentavo \_cCVE, 33  
CapeVerdeanEscudo \_CVE, 33  
CaymanIslandsCent \_cKYD, 33  
CaymanIslandsDollar \_KYD, 33  
CentralAfricanCFACentime \_cXAF, 34  
CentralAfricanCFAFranc \_XAF, 34  
CFPCCentime \_cXPF, 35  
CFPFranc \_XPF, 35  
ChileanCentavo \_cCLP, 25  
ChileanPeso \_CLP, 25  
ChineseRenminbiFen \_cCNY, 25  
ChineseRenminbiYuan \_CNY, 25  
ColombianCentavo \_cCOP, 25  
ColombianPeso \_COP, 25  
ComorianCentime \_cKMF, 33  
Comorianfranc \_KMF, 33  
CongolesCentime \_cCDF, 25  
CongolesFranc \_CDF, 25  
CostaRicanCentimos \_cCRC, 25  
CostaRicanColon \_CRC, 25  
CroatianKuna \_HRK, 26  
CroatianLipa \_cHRK, 26  
CubanCentavo \_cCUP, 32  
CubanoConvertibleCentavo \_cCUC, 32  
CubanoConvertiblePeso \_CUC, 32  
CubanPeso \_CUP, 32  
CzechHaler \_cCZK, 25  
CzechKoruna \_CZK, 25

DanishKrone \_DKK, 25  
 DanishOre \_cDKK, 25  
 DjiboutianCentime \_cDJF, 33  
 DjiboutianFranc \_DJF, 33  
 DominicanCentavo \_cDOP, 25  
 DominicanPeso \_DOP, 25  
  
 EasternCaribbeanCent \_cXCD, 34  
 EasternCaribbeanDollar \_XCD, 34  
 EgyptianPiastre \_cEGP, 25  
 EgyptianPound \_EGP, 25  
 EritreanCent \_cERN, 33  
 EritreanNakfa \_ERN, 33  
 EthiopianBirr \_ETB, 25  
 EthiopianSantim \_cETB, 25  
  
 FalklandIslandsPenny \_cFKP, 33  
 FalklandIslandsPound \_FKP, 33  
 FijianCent \_cFJD, 25  
 FijianDollar \_FJD, 25  
  
 GambianButut \_cGMD, 26  
 GambianDalasi \_GMD, 26  
 GeorgianLari \_GEL, 26  
 GeorgianTetri \_cGEL, 26  
 GhanaianCedi \_GHS, 26  
 GhanaianPesewa \_cGHS, 26  
 GibraltarPenny \_cGIP, 33  
 GibraltarPound \_GIP, 33  
 GuatemalanCentavo \_cGTQ, 26  
 GuatemalanQuetzal \_GTQ, 26  
 GuernseyPenny \_cGGP, 33  
 GuernseyPound \_GGP, 33  
 GuineanCentime \_cGNF, 26  
 GuineanFranc \_GNF, 26  
 GuyaneseCent \_cGYD, 26  
 GuyaneseDollar \_GYD, 26  
  
 HaitianCentime \_cHTG, 26  
 HaitianGourde \_HTG, 26  
 HonduranCentavo \_cHNL, 26  
 HonduranLempira \_HNL, 26  
 HongKongCent \_cHKD, 26  
 HongKongDollar \_HKD, 26  
 HungarianFiller \_cHUF, 26  
 HungarianForint \_HUF, 26

IcelandicKrona \_ISK, 27  
 IndianPaisa \_cINR, 27  
 IndianRupee \_INR, 27  
 IndonesianRupiah \_IDR, 26  
 IndonesianSen \_cIDR, 26  
 IranianRial \_IRR, 27  
 IranianToman \_cIRR, 27  
 IraqiDinar \_IQD, 27  
 IraqiFils \_cIQD, 27  
 IsraeliNewAgora \_cILS, 26  
 IsraeliNewShekel \_ILS, 26  
  
 JamaicanCent \_cJMD, 27  
 JamaicanDollar \_JMD, 27  
 JapaneseYen \_JPY, 27  
 JerseyPenny \_cJEP, 33  
 JerseyPound \_JEP, 33  
 JordanianDinar \_JOD, 33  
 JordanianFils \_cJOD, 33  
  
 KazakhstaniTenge \_KZT, 27  
 KazakhstaniTiyn \_cKZT, 27  
 KenyanCent \_cKES, 27  
 KenyanShilling \_KES, 27  
 KiribatiCent \_cKID, 33  
 KiribatiDollar \_KID, 33  
 KuwaitiDinar \_KWD, 27  
 KuwaitiFils \_cKWD, 27  
 KyrgyzstaniSom \_KGS, 27  
 KyrgyzstaniTiyn \_cKGS, 27  
  
 LaoAtt \_cLAK, 27  
 LaoKip \_LAK, 27  
 LebanesePound \_LBP, 33  
 LebaneseQeresh \_cLBP, 33  
 LiberianCent \_cLRD, 27  
 LiberianDollar \_LRD, 27  
 LibyanDinar \_LYD, 28  
 LibyanDirham \_cLYD, 28  
  
 MacaneseAvo \_cMOP, 33  
 MacanesePataca \_MOP, 33  
 MacedonianDenar \_MKD, 28  
 MacedonianDeni \_cMKD, 28  
 MalagasyAriary \_MGA, 28  
 MalagasyIraimbilanja \_cMGA, 28



MalawianKwacha \_MWK, 28  
 MalawianTambala \_cMWK, 28  
 MalaysianRinggit \_MYR, 28  
 MalaysianSen \_cMYR, 28  
 MaldivianLaari \_cMVR, 28  
 MaldivianRufiyaa \_MVR, 28  
 ManxPenny \_cIMP, 33  
 ManxPound \_IMP, 33  
 MauritanianKhoums \_cMRU, 28  
 MauritanianOuguiya \_MRU, 28  
 MauritianCent \_cMUR, 28  
 MauritianRuppee \_MUR, 28  
 MexicanCentavo \_cMXN, 28  
 MexicanPeso \_MXN, 28  
 MoldovanBan \_cMDL, 28  
 MoldovanLeu \_MDL, 28  
 MongolianMonggo \_cMNT, 28  
 MongolianTogrog \_MNT, 28  
 MoroccanDirham \_MAD, 28  
 MoroccanSantim \_cMAD, 28  
 MozambicanCentavo \_cMZN, 28  
 MozambicanMetical \_MZN, 28  
  
 NamibianCent \_cNAD, 34  
 NamibianDollar \_NAD, 34  
 NepalesePaissa \_cNPR, 34  
 NepaleseRuppee \_NPR, 34  
 NetherlandsAntilleanCent \_cANG, 32  
 NetherlandsAntilleanGuilder \_ANG,  
 32  
 NewTaiwanCent \_cTWD, 31  
 NewTaiwanDollar \_TWD, 31  
 NewZealandCent \_cNZD, 29  
 NewZealandDollar \_NZD, 29  
 NicaraguanCentavo \_cNIO, 29  
 NicaraguanCordoba \_NIO, 29  
 NigerianKobo \_cNGN, 29  
 NigerianNaira \_NGN, 29  
 NorthKoreanChon \_cKPW, 27  
 NorthKoreanWon \_KPW, 27  
 NorwegianKrone \_NOK, 29  
 NorwegianOre \_cNOK, 29  
  
 OmaniBaisa \_cOMR, 34  
 OmaniRial \_OMR, 34  
 PakistaniPaissa \_cPKR, 29  
 PakistaniRuppee \_PKR, 29  
 PanamanianBalboa \_PAB, 34  
 PanamanianCentesimo \_cPAB, 34  
 PapuaNewGuineanKina \_PGK, 29  
 PapuaNewGuineanToea \_cPGK, 29  
 ParaguayanCentimo \_cPYG, 29  
 ParaguayanGuarani \_PYG, 29  
 PennySterling \_cGBP, 26  
 PeruvianCentimo \_cPEN, 29  
 PeruvianSol \_PEN, 29  
 PhilippinePeso \_PHP, 29  
 PhilippineSentimo \_cPHP, 29  
 PolishGrosz \_cPLN, 29  
 PolishZloty \_PLN, 29  
 PoundSterling \_GBP, 26  
  
 QatariDirham \_cQAR, 29  
 QatariRiyal \_QAR, 29  
  
 RomanianBan \_cRON, 29  
 RomanianLeu \_RON, 29  
 RussianKopeyka \_cRUB, 29  
 RussianRuble \_RUB, 29  
 RwandanCentime \_cRWF, 30  
 RwandanFranc \_RWF, 30  
  
 SaintHelenaPenny \_cSHP, 34  
 SaintHelenaPound \_SHP, 34  
 SamoanSene \_cWST, 31  
 SamoanTala \_WST, 31  
 SaoTomeAndPrincipeCentimo  
 \_cSTN, 34  
 SaoTomeAndPrincipeDobra \_STN,  
 34  
 SaudiHalalah \_cSAR, 34  
 SaudiRiyal \_SAR, 34  
 SerbianDinar \_RSD, 29  
 SerbianPara \_cRSD, 29  
 SeychelloisCent \_cSCR, 30  
 SeychelloisRuppee \_SCR, 30  
 SierraLeoneanCent \_cSLL, 30  
 SierraLeoneanLeone \_SLL, 30  
 SingaporeCent \_cSGD, 30  
 SingaporeDollar \_SGD, 30  
 SolomonIslandsCent \_cSBD, 30  
 SolomonIslandsDollar \_SBD, 30

SomalilandCent \_cSQS, 30  
 SomalilandShilling \_SQS, 30  
 SomaliSenti \_cSOS, 30  
 SomaliShilling \_SOS, 30  
 SouthAfricanCent \_cZAR, 31  
 SouthAfricanRand \_ZAR, 31  
 SouthKoreanJeon \_cKRW, 27  
 SouthKoreanWon \_KRW, 27  
 SouthSudanesePiaster \_cSSP, 34  
 SouthSudanesePound \_SSP, 34  
 SriLankanCent \_cLKR, 27  
 SriLankanRupee \_LKR, 27  
 SudanesePound \_SDG, 30  
 SudaneseQirsh \_cSDG, 30  
 SurinameseCent \_cSRD, 30  
 SurinameseDollar \_SRD, 30  
 SwaziCent \_cSZL, 34  
 SwaziLilangeni \_SZL, 34  
 SwedishKrona \_SEK, 30  
 SwedishOre \_cSEK, 30  
 SwissFranc \_CHF, 25  
 SwissRappen \_cCHF, 25  
 SyrianPiastre \_cSYP, 30  
 SyrianPound \_SYP, 30  
  
 TajikistaniDram \_cTJS, 30  
 TajikistaniSamani \_TJS, 30  
 TanzanianSenti \_cTZS, 31  
 TanzanianShilling \_TZS, 31  
 ThaiBaht \_THB, 30  
 ThaiSatang \_cTHB, 30  
 Tonganpaanga \_TOP, 30  
 TonganSeniti \_cTOP, 30  
 TransnistrianKopeck \_cPRB, 34  
 TransnistrianRuble \_PRB, 34  
 TrinidadAndTobagoCent \_cTTD, 31  
 TrinidadAndTobagoDollar \_TTD, 31  
  
 TurkishKurus \_cTRY, 31  
 TurkishLira \_TRY, 31  
 TurkmenistanManat \_TMT, 34  
 TurkmenistanTenge \_cTMT, 34  
 TuvaluanCent \_cTVD, 34  
 TuvaluanDollar \_TVD, 34  
  
 UgandanCent \_cUGX, 31  
 UgandanShilling \_UGX, 31  
 UkrainianHryvnia \_UAH, 31  
 UkrainianKopiyka \_cUAH, 31  
 UnitedArabEmiratesDirham \_AED,  
 32  
 UnitedArabEmiratesFils \_cAED, 32  
 UruguayanCentesimo \_cUYU, 31  
 UruguayanPeso \_UYU, 31  
 USCent \_cUSD, 31  
 USDollar \_USD, 31  
 UzbekistaniSom \_UZS, 31  
 UzbekistaniTiyin \_cUZS, 31  
  
 VenezuelanBolivarSoberano \_VES,  
 31  
 VenezuelanCentimoSoberano \_cVES,  
 31  
 VietnameseDong \_VND, 31  
 VietnameseXu \_cVND, 31  
  
 WestAfricanCFACentime \_cXOF, 35  
 WestAfricanCFAFranc \_XOF, 35  
  
 YemeniDinar \_cYER, 31  
 YemeniRial \_YER, 31  
  
 ZambianKwacha \_ZMW, 31  
 ZambianNgwee \_cZMW, 31  
 ZimbabweanBonds \_ZWL, 35  
 ZimbabweanCent \_cZWL, 35

## Index of Lua Classes and Methods

D.defineBase(symbol, name), 47  
D.new(d), 47  
N.abs(n), 50  
N.acos(n), 53  
N.asin(n), 53  
N.atan(n), 54  
N.cos(n), 52  
N.cosh(q), 54  
N.exp(n), 52  
N.log(n, base=nil), 51  
N.mean(n), 50  
N.new(n=nil), 49  
N.new(str), 49  
N.new(x, dx=nil), 49  
N.sin(n), 52  
N.sinh(q), 54  
N.sqrt(n), 51  
N.tan(n), 53  
N.tanh(q), 55  
N.uncertainty(n), 50  
Q.abs(q), 40  
Q.acos(q), 43  
Q.acosh(q), 45, 56  
Q.addPrefix(prefixes, units),  
    38  
Q.asin(q), 43  
Q.asinh(q), 45, 55  
Q.atan(q), 43  
Q.atanh(q), 45, 56  
Q.cos(q), 42  
Q.cosh(q), 44  
Q.define(symbol, name, q), 37  
Q.defineBase(symbol, name, dimension),  
    36  
Q.definePrefix(symbol, name, factor),  
    38  
Q.exp(q), 41  
Q.isclose(self, q, r), 38  
Q.log(q, base=nil), 41  
Q.max(q1, q2, ...), 40  
Q.min(q1, q2, ...), 40  
Q.new(q=nil), 36  
Q.sin(q), 42  
Q.sinh(q), 44  
Q.sqrt(q), 41  
Q.tan(q), 42  
Q.tanh(q), 44  
Q.to(self, q=nil), 39  
Q.tosiunitx(self, param, mode=Q.siunitx\_SI),  
    39  
Unit.new(symbol, name,  
    prefixsymbol=nil,  
    prefixname=nil), 48  
Unit.new(u=nil), 48  
Unit.tosiunitx(self), 48