بسم الله الرحمن الرحيم

Quantity System Specifications

Contents

[Introduction 1](#_Toc240669059)

[Quantities 2](#_Toc240669060)

[Units 2](#_Toc240669061)

[Unit information 3](#_Toc240669062)

[Quantity Types 3](#_Toc240669063)

[Qs Values 4](#_Toc240669064)

[Scalar 4](#_Toc240669065)

[Addition '+' and Subtraction '-' 4](#_Toc240669066)

[Multiplication '\*' and Division '/' Operations 6](#_Toc240669067)

[Power '^' Operation 7](#_Toc240669068)

[Scalar Factorial '!' Operation 8](#_Toc240669069)

[Scalar Absolute |x| 8](#_Toc240669070)

[Vector 9](#_Toc240669071)

[Declarations 9](#_Toc240669072)

[Vector Scalar Operations 9](#_Toc240669073)

[Vector Multiplication '\*', and Division 9](#_Toc240669074)

[Vector Dot Product '.' 9](#_Toc240669075)

[Vector Cross Product 'x' 10](#_Toc240669076)

[Vector Magnitude ||x|| 10](#_Toc240669077)

[Matrix 11](#_Toc240669078)

[Declaration 11](#_Toc240669079)

[Matrix Scalar Operations '\*', '/', '+', '-' 11](#_Toc240669080)

[Matrix Ordinary Multiplication '\*' 12](#_Toc240669081)

[Matrix element multiplication and power '.', and '^.' 12](#_Toc240669082)

[Determinant |x| 13](#_Toc240669083)

[Functions 13](#_Toc240669084)

[Declaration 13](#_Toc240669085)

[Passing function as arguments 14](#_Toc240669086)

[Sequence 15](#_Toc240669087)

[Declaration 15](#_Toc240669088)

[Sequence Elements 15](#_Toc240669089)

[Access Sequence Elements 15](#_Toc240669090)

[Set Sequence Elements 16](#_Toc240669091)

[Series Operator [n++m] 16](#_Toc240669092)

[Product Operator [n\*\*m] 17](#_Toc240669093)

[Average [n!!m] 17](#_Toc240669094)

[Range [n..m] 18](#_Toc240669095)

[Appendix 19](#_Toc240669096)

[Qs Operators 19](#_Toc240669097)

[List of Quantities 19](#_Toc240669098)

[List of Units 21](#_Toc240669099)

# Introduction

For many decades humanity had to deal with numbers in many forms, from the representation by sticks till the numbering systems that are common now.

This includes the decimal numbering system and the other specific numbering systems that are known to the computer science students.

Reaching the current era we had a ton of proven numbering concepts. Human on its life time has developed many concepts and methods that resulted in algebra, calculus, and algorithms.

In the decades before computers appearance people used to express the numbers followed by a unit that describe what is this number referring to.

Scientist used to call naked numbers as a magnitude and the unit as if it were its class.

Then calling the whole thing as a quantity.

So it is 10 feets, and 7 grams, or 5 cubits.

The units have been evolved due to the progression of the humanity itself. It has a many revisions and remaking and also inventions and new units have been introduced in every decade to express what are the new quantities and what they should be.

In this document I am setting the specifications of the Quantity System Calculator ('Qs' from now on) that shipped as a direct application for the Quantity System Framework.

The Quantity System Framework is an attempt to express a higher modeling expression for the programming languages variables to be a quantity rather than a dimensionless quantity all the time.

# Quantities

Quantities are a higher representation for numbers, rather than saying Two as a number, you say Two Oranges. This is a quantity of Oranges. It may be apple, peach, or in an abstract level it is a quantity of fruits.

In the same conclusion we can say 10 meters, feet, or inches that all refer to the same quantity which is 'Length'

Quantities are found in every place around our life, Speed, Acceleration, Angle, or Torque.

As long the quantities are similar then we can sum, or subtract them without problems. However multiplication and division of quantities result in other quantities.

Like the speed divided by time, the resulted quantity is the acceleration.

In the Qs any summation or subtraction for any different units that refer to the same quantities is a valid operation.

The syntax of the quantity declaration is number followed by unit between '<' '>' brackets.

So 30<m> + 20<ft> - 10<in> is a valid operation



## Units

Units are the representation of Quantities magnitudes but in the form of understandable letters.

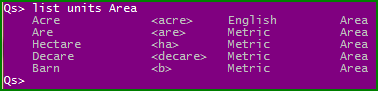
The two famous quantities that all people deal with are Length and Area

Let's see what the Qs have a hard coded units for those two quantities

Length:

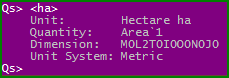


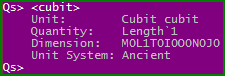
Area:



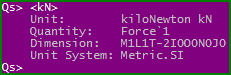
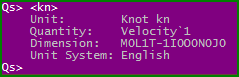
### Unit information

To know information about any unit write the unit between less than '<' and bigger '>' than brackets.





Units are case sensitive which means that <kn> is different than <kN> and <S> is different than <s>.





### Quantity Types

There are only two classes of quantities, either dimensionless or non dimensionless quantities.

#### Dimensionless Quantities

Writing a number without any units is like writing a dimensionless quantity.

The dimensionless quantity unit is <1>.



#### Non Dimensionless Quantities

To write a number with a unit simply right the number and the unit followed by it

i.e. 20<m> or 40<kg>



As you can notice the Qs is predicting the quantity from the unit you give it.

# Qs Values

The Qs currently is supporting 3 types of values you can work with Scalar, Vector, and Matrix.

In future releases it will support Tensors as well.

## Scalar

Scalar expressing any quantity the like 50<ft> or 20<J>

### Addition '+' and Subtraction '-'

Addition or Subtractions of scalars only succeed if the quantities used are the same quantities

For example Energy (Work) cannot be added to Acceleration.

However the Qs calculator can predict the conversions factor between units.

Consider a sample with feet, inch, meter, and kilo meter units

The Qs will add all the quantities and return the value with the same unit of the first scalar in the operation



With this facility, let's have some fun with units by asking our selves

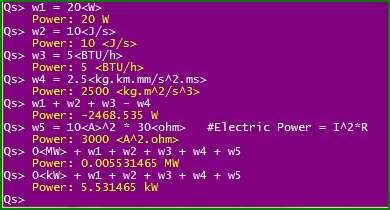
What is the speed of yard per hour <yd/h> from 30 meter per second <m/s>?



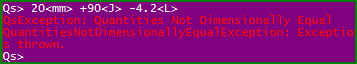
The Qs can handle many units' conversions and it is intelligent enough to make the right conversion for you.

Consider the units of Power Quantity for example

* Watt <W>
* Joule per second <J/s>
* BTU (British Thermal Unit) Per hour <BTU/h>
* More radical formation of units <kg.km.mm/s^2.ms> which is kilogram \* kilo meter \* milli meter per square second \* milli second.



Whereas those operations fail in the different quantities scenarios



### Multiplication '\*' and Division '/' Operations

The multiplication and division in Qs are very interesting because a quantity prediction is occurred whenever you multiply or divide quantities.

If the resulted quantity is known by the Qs it will be shown to you as a result.

For this feature you can always make your calculations and check for the last answer quantity.

If the quantity what you would expect then your calculations doesn't have a logical error, but if you found something that don't fit with your expectations then you have to review your calculations from the start.

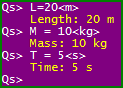
By this feature I am hoping that calculations and scientific calculations go smoothly and without errors.

For example let's start with this scenario

L = 20<m>

M =10<kg>

T = 5<s>



Dividing length over a time will produce a Velocity (Speed).



Dividing Velocity over time again produce Acceleration.



And by multiplying Acceleration with mass we obtain Force.



What if you want to obtain the result with kilo Newton?

Simply you can add the result value to 0<kN>

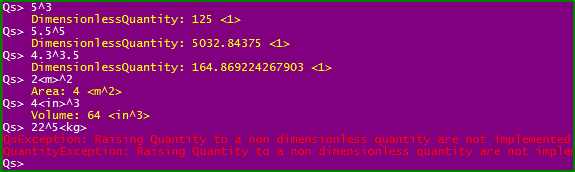


### Power '^' Operation

You can even raise the quantity to a power

However the power should be dimensionless quantity

Also the power can be from the real set of numbers.

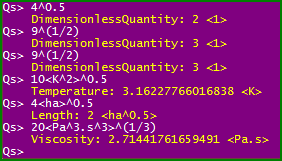


#### Roots

With a power number from 0 to 1 is like having the root of the number

For example the square root of 4 is 2 and expressed as 4^0.5 or 9^(1/2)

It even gives you the length from area unit like 4<ha>^0.5



### Scalar Factorial '!' Operation

The factorial '!' operator is used to get the factorial of the quantity and it is a postfix operator means it affects only the quantity declared before it.



And the result is predicted for the corresponding quantity if found.





Factorial also has a nice feature that it supports fractional numbers



### Scalar Absolute |x|

You can put your scalar or the variable that express scalars between two vertical bars to get the absolute quantity.



## Vector

The vector in quantity have some unique features, it has three types of multiplication operators, and a magnitude operator similar as the mathematics operator.

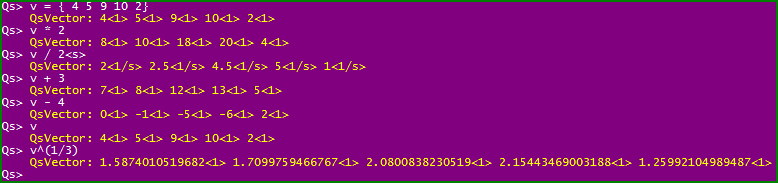
### Declarations

Vector is written between two curly brackets '{' '}' and the quantities are separated by comma ',' or space



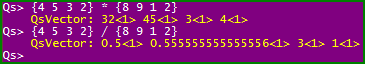
### Vector Scalar Operations

The vector can have all basic operations with scalar '+', '-', '\*', '/', '^'.



### Vector Multiplication '\*', and Division

Normal multiplication operator for normal multiplication of vectors



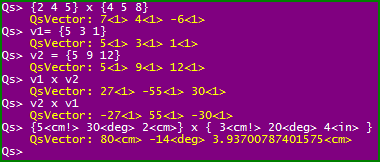
### Vector Dot Product '.'

Use the '.' letter as a dot product operator to get the dot product of vectors.



### Vector Cross Product 'x'

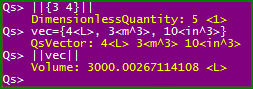
Use the 'x' letter as a cross product operator to get the vector product.



Note: the cross product only defined for three component vectors.

### Vector Magnitude ||x||

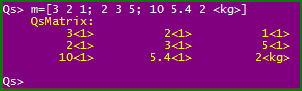
You can get the magnitude of the vector by enclosing it between two vertical bars.



## Matrix

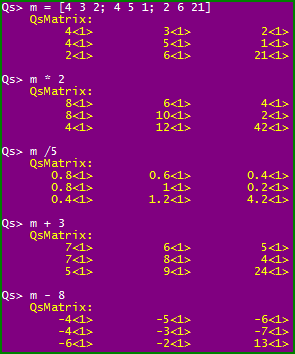
### Declaration

The matrix declaration is similar to matlab matrix declaration.



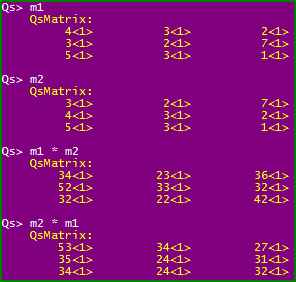
And of course because our scalar is a quantity so you can write your quantities with any units you choose.

### Matrix Scalar Operations '\*', '/', '+', '-'



### Matrix Ordinary Multiplication '\*'

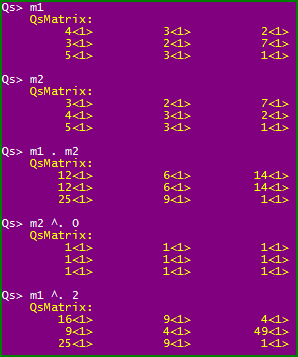
The ordinary matrix in matrix multiplication is made by applying the multiplication operator '\*'.



### Matrix element multiplication and power '.', and '^.'

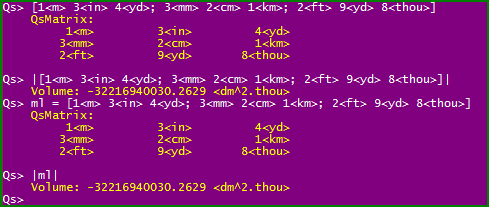
Use the '.' dot operator to get the multiplication by elements of matrix also you can get the power of matrix elements by using '^.' .

Note: Element Power operator '^.' Is different than the Array power operator in matlab '.^'.



### Determinant |x|

If you have declared a matrix which is 2x2 or 3x3 you can get its determinant by enclosing it between two vertical bars.



Note: yes the absolute operator is the same as the determinant operator and the Qs is intelligent enough to differentiate between scalar or matrix.

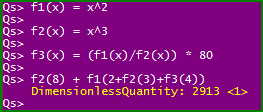
# Functions

In designing the function syntax in Qs I intended to let it as simple as I can so any one with a little math knowledge can write the function as he can see.

## Declaration

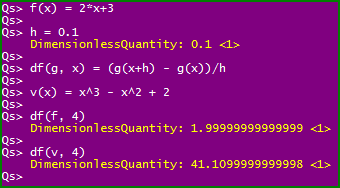
[Function Name]([arg1, arg2, …, arg11]) = function body

The arguments of function are limited to 11 arguments which I think they are more than enough.



## Passing function as arguments

Functions in Qs can be sent as argument to another function. This way you can define simple functions that make numerical derivatives.



# Sequence

Sequence is a set of numbers that could be accessed by index.

There is a famous sequence that called Fibonacci sequence.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …

## Declaration

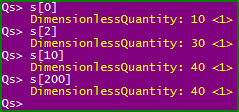
To declare a sequence in Qs use the '..>' operator and write elements after it separated by semi colons.



## Sequence Elements

### Access Sequence Elements

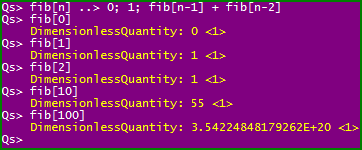
To access the elements of the sequence write the element number between squared brackets



Sequence is behaving like arrays in this manner but with very special feature. Sequence in Qs is infinite. When access any element of the sequence the Qs always evaluate the last element of the sequence to get its value.

By this feature you guarantee that you always have a number in the end of the element access.

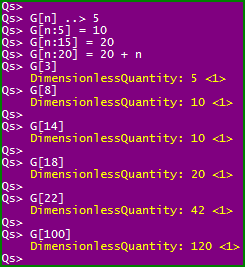
For example take the Fibonacci sequence as an example. When you declare the Fibonacci series you only declare three elements and the other elements will be calclulated automatically from the last element.



### Set Sequence Elements

The intelligent design of the sequence permit you to set specific elements of sequence after declaration.

You can set successive elements or elements that are not neighbors. For example you can make your sequence return 5 for the first 5 elements then 10 for the next 10 elements, then you may make a calculation element in the tail of the sequence.



So this means that when accessing the element it looks for its value first to evaluate, if it is empty then it looks to previous elements until it encounters a non empty element then evaluate its value.

## Series Operator [n++m]

Series is a summation of the sequence elements from index to index.

Suppose that you want to define the Exponential series to be used in the program.

The exponential series mathematically is as follows.

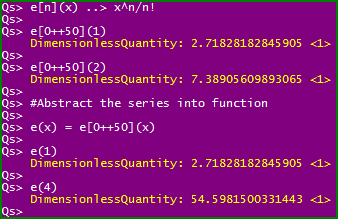
To make this series in Qs you first define the sequence with one parameter x

e[n](x) ..> x^n/n!

And then call the summation operator inside the indexer of the sequence.

e[0++50](1)

you can notice that this sequence accept beside its indexer 'n' a parameter called 'x'

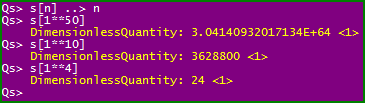


for abstracting the series call you can put the series call into function

e(x) = e[0++50](x)

## Product Operator [n\*\*m]

The concept of sequence can also be reused to make multiplication of elements



## Average [n!!m]

To support more aggressive feature I thought that adding mean value for the sequence is something that makes it new and also illustrate the ability to increase sequence operators in the future.

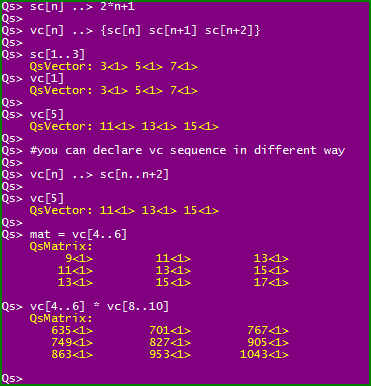


## Range [n..m]

Range operator used to return a vector from scalar elements sequence or matrix from vector elements sequence.

This nice feature I couldn't know what it will be used for

But I think it is nice to have feature.



# Appendix

## Qs Operators

|  |  |  |
| --- | --- | --- |
| Symbol | Operator Name | Apply to |
| + | Addition | Scalar, Vector, Matrix |
| - | Subtraction | Scalar, Vector, Matrix |
| \* | Multiply | Scalar, Vector, Matrix |
| ^ | Power | Scalar, Vector, Matrix |
| . | Dot Product | Scalar, Vector, Matrix |
| ^. | Dot Power | Scalar, Vector, Matrix |
| x | Cross Product (x letter) | Scalar, Vector |
| ^x | Cross Power | Scalar, Vector |
| / | Division | Scalar, Vector, Matrix |
| ! | Postfix factorial | Scalar, Vector, Matrix |
| | value | | Absolute, Determinant | Scalar, Matrix |
| || value || | Magnitude | Vector |
| ..> | Positive Sequence | Sequence |
| <.. | Negative Sequence | Sequence |
| S[n++m] | Sequence Series operator | Sequence |
| S[n\*\*m] | Sequence Product operator | Sequence |
| S[n!!m] | Sequence Average operator | Sequence |
| S[n..m] | Sequence Range operator | Sequence |

## List of Quantities

M: Mass

L: Length

T: Time

I: Current

O: Temperature

N: Amount of Substance

J: Luminous Intensity

|  |  |  |
| --- | --- | --- |
|  | Quantity | Dimension |
| 1 | Acceleration | M0L1T-2I0O0N0J0 |
| 2 | AmountOfSubstance | M0L0T0I0O0N1J0 |
| 3 | Angle | M0L0T0I0O0N0J0 |
| 4 | AngularAcceleration | M0L0T-2I0O0N0J0 |
| 5 | AngularMomentum | M1L2T-1I0O0N0J0 |
| 6 | AngularVelocity | M0L0T-1I0O0N0J0 |
| 7 | Area | M0L2T0I0O0N0J0 |
| 8 | Capacitance | M-1L-2T4I2O0N0J0 |
| 9 | CatalyticActivity | M0L0T-1I0O0N1J0 |
| 10 | Density | M1L-3T0I0O0N0J0 |
| 11 | DimensionlessQuantity | M0L0T0I0O0N0J0 |
| 12 | ElectricalCurrent | M0L0T0I1O0N0J0 |
| 13 | ElectricCharge | M0L0T1I1O0N0J0 |
| 14 | ElectricConductance | M-1L-2T3I2O0N0J0 |
| 15 | ElectricResistance | M1L2T-3I-2O0N0J0 |
| 16 | ElectromotiveForce | M1L2T-3I-1O0N0J0 |
| 17 | Energy | M1L2T-2I0O0N0J0 |
| 18 | Force | M1L1T-2I0O0N0J0 |
| 19 | Frequency | M0L0T-1I0O0N0J0 |
| 20 | HeatCapacity | M1L2T-2I0O-1N0J0 |
| 21 | Illuminance | M0L-2T0I0O0N0J1 |
| 22 | Inductance | M1L2T-2I-2O0N0J0 |
| 23 | KinematicViscosity | M0L2T-1I0O0N0J0 |
| 24 | Length | M0L1T0I0O0N0J0 |
| 25 | LuminousFlux | M0L0T0I0O0N0J1 |
| 26 | LuminousIntensity | M0L0T0I0O0N0J1 |
| 27 | MagneticFlux | M1L2T-2I-1O0N0J0 |
| 28 | MagneticFluxDensity | M1L0T-2I-1O0N0J0 |
| 29 | Mass | M1L0T0I0O0N0J0 |
| 30 | MassMomentOfInertia | M1L2T0I0O0N0J0 |
| 31 | MolarMass | M1L0T0I0O0N-1J0 |
| 32 | Momentum | M1L1T-1I0O0N0J0 |
| 33 | Power | M1L2T-3I0O0N0J0 |
| 34 | Pressure | M1L-1T-2I0O0N0J0 |
| 35 | RadiantIntensity | M1L2T-3I0O0N0J0 |
| 36 | RadiusLength | M0L1T0I0O0N0J0 |
| 37 | RotationalStiffness | M1L2T-2I0O0N0J0 |
| 38 | SolidAngle | M0L0T0I0O0N0J0 |
| 39 | SpecificHeat | M0L2T-2I0O-1N0J0 |
| 40 | SpecificVolume | M-1L3T0I0O0N0J0 |
| 41 | SpecificWeight | M1L-2T-2I0O0N0J0 |
| 42 | Stiffness | M1L0T-2I0O0N0J0 |
| 43 | Temperature | M0L0T0I0O1N0J0 |
| 44 | Time | M0L0T1I0O0N0J0 |
| 45 | Torque | M1L2T-2I0O0N0J0 |
| 46 | Velocity | M0L1T-1I0O0N0J0 |
| 47 | Viscosity | M1L-1T-1I0O0N0J0 |
| 48 | Volume | M0L3T0I0O0N0J0 |
| 49 | VolumeFlowRate | M0L3T-1I0O0N0J0 |

## List of Units

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Unit Name | Symbol | System | Quantity Name |
| 1 | Mole | <mol> | Metric.SI | AmountOfSubstance |
| 2 | PoundMole | <lbmol> | English | AmountOfSubstance |
| 3 | Revolution | <r> | Misc | Angle |
| 4 | Gradian | <grad> | Misc | Angle |
| 5 | ArcDegree | <deg> | Misc | Angle |
| 6 | ArcMinute | <arcmin> | Misc | Angle |
| 7 | ArcSecond | <arcsec> | Misc | Angle |
| 8 | MilliArcSecond | <mas> | Misc | Angle |
| 9 | Radian | <rad> | Metric.SI | Angle |
| 10 | RevolutionPerMinute | <rpm> | Misc | AngularVelocity |
| 11 | Acre | <acre> | English | Area |
| 12 | Are | <are> | Metric | Area |
| 13 | Hectare | <ha> | Metric | Area |
| 14 | Decare | <decare> | Metric | Area |
| 15 | Barn | <b> | Metric | Area |
| 16 | Farad | <F> | Metric.SI | Capacitance |
| 17 | Katal | <kat> | Metric.SI | CatalyticActivity |
| 18 | Ampere | <A> | Metric.SI | ElectricalCurrent |
| 19 | Coulomb | <C> | Metric.SI | ElectricCharge |
| 20 | Siemens | <S> | Metric.SI | ElectricConductance |
| 21 | Ohm | <ohm> | Metric.SI | ElectricResistance |
| 22 | Volt | <V> | Metric.SI | ElectromotiveForce |
| 23 | Erg | <erg> | Metric.Cgs | Energy |
| 24 | Joule | <J> | Metric.SI | Energy |
| 25 | ElectronVolt | <eV> | Natural | Energy |
| 26 | BTU | <BTU> | English | Energy |
| 27 | GramForce | <gf> | Metric.Gravitational | Force |
| 28 | Pond | <p> | Metric.Gravitational | Force |
| 29 | Sthène | <sn> | Metric.Mts | Force |
| 30 | Newton | <N> | Metric.SI | Force |
| 31 | PoundForce | <lbf> | English | Force |
| 32 | Poundal | <pdl> | English | Force |
| 33 | Dyne | <dyn> | Metric.Cgs | Force |
| 34 | Hertz | <Hz> | Metric.SI | Frequency |
| 35 | Lux | <lx> | Metric.SI | Illuminance |
| 36 | Henry | <H> | Metric.SI | Inductance |
| 37 | Stokes | <St> | Metric.Cgs | KinematicViscosity |
| 38 | AstronomicalUnit | <au> | Astronomical | Length |
| 39 | LightYear | <ly> | Astronomical | Length |
| 40 | Parsec | <pc> | Astronomical | Length |
| 41 | Cubit | <cubit> | Ancient | Length |
| 42 | Thou | <thou> | English | Length |
| 43 | Inch | <in> | English | Length |
| 44 | Foot | <ft> | English | Length |
| 45 | Yard | <yd> | English | Length |
| 46 | Fathom | <ftm> | English | Length |
| 47 | Furlong | <fur> | English | Length |
| 48 | Mile | <mil> | English | Length |
| 49 | League | <league> | English | Length |
| 50 | Cable | <cable> | English | Length |
| 51 | NauticalMile | <nmi> | English | Length |
| 52 | Link | <lnk> | English | Length |
| 53 | Rod | <rod> | English | Length |
| 54 | Chain | <chain> | English | Length |
| 55 | Metre | <m> | Metric.SI | Length |
| 56 | Angstrom | <an> | Misc | Length |
| 57 | Lumen | <lm> | Metric.SI | LuminousFlux |
| 58 | Candela | <cd> | Metric.SI | LuminousIntensity |
| 59 | Weber | <Wb> | Metric.SI | MagneticFlux |
| 60 | Tesla | <T> | Metric.SI | MagneticFluxDensity |
| 61 | Hyl | <hyl> | Metric.Gravitational | Mass |
| 62 | ElectronMass | <me> | Natural | Mass |
| 63 | ProtonMass | <mp> | Natural | Mass |
| 64 | Dalton | <Da> | Natural | Mass |
| 65 | UnifiedAtomicMass | <u> | Natural | Mass |
| 66 | MetricTonne | <mt> | Metric.Mts | Mass |
| 67 | Gram | <g> | Metric.SI | Mass |
| 68 | Slug | <slug> | English | Mass |
| 69 | Grain | <gr> | English | Mass |
| 70 | Drachm | <drachm> | English | Mass |
| 71 | Ounce | <oz> | English | Mass |
| 72 | Pound | <lbm> | English | Mass |
| 73 | Stone | <st> | English | Mass |
| 74 | Quarter | <quarter> | English | Mass |
| 75 | Hundredweight | <cwt> | English | Mass |
| 76 | Tonne | <t> | English | Mass |
| 77 | SolarMass | <Mo> | Astronomical | Mass |
| 78 | Watt | <W> | Metric.SI | Power |
| 79 | HorsePower | <HP> | English | Power |
| 80 | Psi | <psi> | English | Pressure |
| 81 | Bar | <bar> | Metric | Pressure |
| 82 | Barye | <Ba> | Metric.Cgs | Pressure |
| 83 | Pascal | <Pa> | Metric.SI | Pressure |
| 84 | Pièze | <pz> | Metric.Mts | Pressure |
| 85 | Steradian | <sr> | Metric.SI | SolidAngle |
| 86 | Kelvin | <K> | Metric.SI | Temperature |
| 87 | Celsius | <°C> | Metric | Temperature |
| 88 | Rankine | <R> | English | Temperature |
| 89 | Fahrenheit | <°F> | English | Temperature |
| 90 | Second | <s> | Shared | Time |
| 91 | Minute | <min> | Misc | Time |
| 92 | Hour | <h> | Misc | Time |
| 93 | Day | <d> | Misc | Time |
| 94 | JulianYear | <a> | Astronomical | Time |
| 95 | LightSpeed | <c0> | Natural | Velocity |
| 96 | Knot | <kn> | English | Velocity |
| 97 | Poise | <P> | Metric.Cgs | Viscosity |
| 98 | CubicCentimetre | <cc> | Misc | Volume |
| 99 | LiquidPint | <lpt> | English.US | Volume |
| 100 | DryPint | <dpt> | English.US | Volume |
| 101 | DryGallon | <dgal> | English.US | Volume |
| 102 | DryQuart | <dqt> | English.US | Volume |
| 103 | Stère | <stère> | Metric.Mts | Volume |
| 104 | FluidOunce | <fl\_oz> | English | Volume |
| 105 | Gill | <gill> | English | Volume |
| 106 | Cup | <cup> | English | Volume |
| 107 | Pint | <pt> | English | Volume |
| 108 | Quart | <qt> | English | Volume |
| 109 | Gallon | <gal> | English | Volume |
| 110 | Litre | <L> | Metric | Volume |
| 111 | CubicFeetPerMinute | <cfm> | English | VolumeFlowRate |
| 112 | GallonPerMinute | <gpm> | English | VolumeFlowRate |