FOUND HERE :
https://chieftalk.chiefarchitect.com/topic/25611-macro-returning-units/?tab=comments#comment-205770

These are the apparent OOB settings...
unit = Not set
show_unit = true
show_leading_zero = true
show_trailing_zeros = false
use_fractions = false
decimal_places = 6
denominator = 16
thousands_separator = ,
show_denominator = true
reduce_fractions = gcd (*greatest common divisor)*

I think that all of the True | False settings are defaulted to False

An example : area.to_sq_m.round(2)

It's not a full list if what can be done, an explanation of exactly how the conversions work, or an explanation of the different types of Measurements (Linear vs. Area vs. Volume), but here's a list of a few of the basic methods you can use to convert measurements to floats based on other units:

.to_inch	.to_sq_inch	.to_cu_inch
.to_in	.to_sq_in	.to_cu_in
.to_foot	.to_sq_foot	.to_cu_foot
.to_ft	.to_sq_ft	.to_cu_ft
.to_yard	.to_sq_yard	.to_cu_yard
.to_yd	.to_sq_yd	.to_cu_yd
.to_mm	.to_sq_mm	.to_cu_mm
.to_cm	.to_sq_cm	.to_cu_cm
.to_dm	.to_sq_dm	.to_cu_dm
.to_m	.to_sq_m	.to_cu_m

An example : area.to_sq_m.round(2)

And here's a list of what you can use to convert a float to a measurement...

.inch	.sq_in	.cu_foot
.in	.sq_foot	.cu_ft
.foot	.sq_ft	.cu_yard
.ft	.sq_yard	.cu_yd
.yard	.sq_yd	
.yd	.sq_mm	.cu_mm
.mm	.sq_cm	.cu_cm
.cm	.sq_dm	.cu_dm
.dm	.sq_m	.cu_m
.m	.cu_inch	
.sq_inch	.cu_in	

...again, learning exactly how the conversion works is a bit more complicated but if you use .to_s you will get the newly created Measurement and its units.

There are other similar methods as well such as .convert_to (as was mentioned above by Ben) as well as Measurement.new(value, optional unit).

In addition Chief also has a built in NumberFormatter functionality that you can use to format various measurements. It basically works exactly like the Dimension formatting options we have. I don't have the time or inclination to go into all of it in this post, but it's pretty cool.

Notably missing from the Help is an example of using the NumberFormatter class. Basically it has to be created first. There's no reason to create it for every time you want to format a number - particularly if you are always going to be using the same parameters.

Here's the way I create a persistent instance of the class and how it can be used:

- $NF = NumberFormatter.new
- $NF.unit = "'-\""
- $NF.use_fractions = true
- $NF.denominator = 8

This sets the parameters to what I want. Then I can use the following whenever I need to format a number:

- $NF.apply(*123.5.in*) ---> 10' 3-1/2"

Note the inclusion of .in so the formatter knows the value is in inches. If passing a Measurement value then that wouldn't need to be included.

```
$NF = NumberFormatter.new

$NF.unit = '\'-"'

$NF.use_fractions = true

def fi(n)

$NF.apply(n)

end

''                                  # " blocks displaying result in labels
```

Variables that start with a Capitol letter are Constant Variables but I am not sure how different they are than Global Variables but they do work outside of their macro unlike Local Variables with a Lower Case letter.

One of the unique behaviors of a Constant over a Global Variable is that Ruby issues a warning if we try to alter the value of the Constant. This behavior doesn't really seem to be usable if we're working outside the Ruby console though.