Inhaltsverzeichnis

1. Modul:Val 2
2. Modul:Val/units 16

Modul:Val

This module implements Vorlage:Tl.

The following modules are developed:

- Module:Val • Main module.
- Module:Val/units • Definitions for units built-in to val.

Use Vorlage:Tl for testing, for example:

- `{{val/sandbox|1234.5678|(23)|u=cm}}` → Vorlage:Val/sandbox
- `{{val/sandbox|1234.5678|1.23|u=cm}}` → Vorlage:Val/sandbox
- `{{val/sandbox|1234.5678|1.23|4.56|u=cm}}` → Vorlage:Val/sandbox
- `{{val/sandbox|1234.5678|e=3|u=cm}}` → Vorlage:Val/sandbox
- `{{val/sandbox|1234.5678|(23)|e=3|u=cm}}` → Vorlage:Val/sandbox
- `{{val/sandbox|1234.5678|1.23|e=3|u=cm}}` → Vorlage:Val/sandbox
- `{{val/sandbox|1234.5678|1.23|4.56|e=3|u=cm}}` → Vorlage:Val/sandbox
- `{{val/sandbox|1234.5678|1.23|4.56|e=3|u=cm|end=$|+errend=U$|-errend=L$}}` → Vorlage:Val/sandbox
- `{{val/sandbox|1234.5678|(23)|u=deg}}` → Vorlage:Val/sandbox
- `{{val/sandbox|1234.5678|1.23|u=deg}}` → Vorlage:Val/sandbox
- `{{val/sandbox|1234.5678|1.23|4.56|u=deg}}` → Vorlage:Val/sandbox
- `{{val/sandbox|1234.5678|e=3|u=deg}}` → Vorlage:Val/sandbox
- `{{val/sandbox|1234.5678|(23)|e=3|u=deg}}` → Vorlage:Val/sandbox
- `{{val/sandbox|1234.5678|1.23|e=3|u=deg}}` → Vorlage:Val/sandbox
- `{{val/sandbox|1234.5678|1.23|4.56|e=3|u=deg}}` → Vorlage:Val/sandbox
- `{{val/sandbox|1234.5678|1.23|4.56|e=3|u=deg|end=$|+errend=U$|-errend=L$}}` → Vorlage:Val/sandbox

```
-- For Template:Val, output a number and optional unit.
-- Format options include scientific and uncertainty notations.

local data_module = 'Module:Val/units'
local convert_module = 'Module:Convert'

local function valerror(msg, nocat, iswarning)
        -- Return formatted message text for an error or warning.
        -- Can append "#FormattingError" to URL of a page with a problem to find
        local anchor = '<span id="FormattingError"></span>'
        local body, category
        if nocat or mw.title.getCurrentTitle():inNamespaces(1, 2, 3, 5) then
                -- No category in Talk, User, User_talk, or Wikipedia_talk.
                category = ''
        else
                category = '[[Category:Pages with incorrect formatting templates
        end
        iswarning = false  -- problems are infrequent so try showing large error
        if iswarning then
```

```
                body = '<sup class="noprint Inline-Template" style="white-space:r
                        '[[Template:Val|<span title="' ..
                        msg:gsub('"', '&quot;') ..
                        '">warning</span>]]</sup>'
        else
                body = '<strong class="error">' ..
                        'Error in &#123;&#123;[[Template:val|val]]&#125;&#125;: '
                        msg ..
                        '</strong>'
        end
        return anchor .. body .. category
end

local range_types = {
        -- No need for ' ' because nowrap applies to all output.
        [","]   = ", ",
        ["by"]  = " by ",
        ["-"]   = "–",
        ["–"]   = "–",
        ["and"] = " and ",
        ["or"]  = " or " ,
        ["to"]  = " to " ,
        ["x"]   = " × ",
        ["×"]   = " × ",
        ["/"]   = "/",
}
local range_repeat_unit = {
        -- WP:UNIT wants unit repeated when a "multiply" range is used.
        ["x"]   = true,
        ["×"]   = true,
}

local function extract_item(index, numbers, arg)
        -- Extract an item from arg and store the result in numbers[index].
        -- If no argument or if argument is valid, return nil (no error);
        -- otherwise, return an error message.
        -- The stored result is:
        -- * a table for a number (empty if there was no specified number); or
        -- * a string for range text
        -- Input like 1e3 is regarded as invalid for all except argument 1
        -- which accepts e notation as an alternative to the 'e' argument.
        -- Input commas are removed so 1,234 is the same as 1234.
        local which = index
        local function fail(msg)
                local description
                if which == 'e' then
                        description = 'exponent parameter (<b>e</b>)'
                else
                        description = 'parameter ' .. which
                end
                return description .. ' ' .. (msg or 'is not a valid number') ..
        end
        local result = {}
        local range = range_types[arg]
        if range then
                if type(index) == 'number' and (index % 2 == 0) then
                        if index == 2 then
                                if numbers[1] and numbers[1].exp then
                                        return fail('cannot use a range if the fi
                                end
                                numbers.has_ranges = true
                        else
                                if not numbers.has_ranges then
                                        return fail('needs a range in parameter 2
```

```
                                end
                        end
                        numbers[index] = range
                        if range_repeat_unit[arg] then
                                -- Any "repeat" range forces unit (if any) to be
                                numbers.isrepeat = true
                        end
                        return nil
                end
                return fail('does not accept a range')
        end
        if numbers.has_ranges and type(index) == 'number' and (index % 2 == 0) th
                return fail('should be a range')
        end
        if index == 'e' then
                local e = numbers[1] and numbers[1].exp
                if e then
                        if arg then
                                return fail('cannot be used if the first paramete
                        end
                        arg = e
                        which = 1
                end
        end
        if arg and arg ~= '' then
                arg = arg:gsub(',', '')
                if arg:sub(1, 1) == '(' and arg:sub(-1) == ')' then
                        result.parens = true
                        arg = arg:sub(2, -2)
                end
                local a, b = arg:match('^(.+)[Ee](.+)$')
                if a then
                        if index == 1 then
                                arg = a
                                result.exp = b
                        else
                                return fail('cannot use e notation')
                        end
                end
                local isnegative, propersign, prefix
                local minus = '−'
                prefix, arg = arg:match('^(.-)([%d.]+)$')
                local value = tonumber(arg)
                if not value then
                        return fail()
                end
                if arg:sub(1, 1) == '.' then
                        arg = '0' .. arg
                end
                if prefix == '' then
                        -- Ignore.
                elseif prefix == '±' then
                        -- Display for first number, ignore for others.
                        if index == 1 then
                                propersign = '±'
                        end
                elseif prefix == '+' then
                        propersign = '+'
                elseif prefix == '-' or prefix == minus then
                        propersign = minus
                        isnegative = true
                else
                        return fail()
                end
```

```
                result.clean = arg
                result.sign = propersign or ''
                result.value = isnegative and -value or value
        end
        numbers[index] = result
        return nil  -- no error
end

local function get_args(numbers, args)
        -- Extract arguments and store the results in numbers.
        -- Return nothing (no error) if ok; otherwise, return an error message.
        for index = 1, 99 do
                local which = index
                local arg = args[which]  -- has been trimmed
                if not arg then
                        which = 'e'
                        arg = args[which]
                end
                local msg = extract_item(which, numbers, arg)
                if msg then
                        return msg
                end
                if which == 'e' then
                        break
                end
                if index > 19 then
                        return 'too many parameters'
                end
        end
        if numbers.has_ranges and (#numbers % 2 == 0) then
                return 'need a number after the last parameter because it is a ra
        end
end

local function get_scale(text, ucode)
        -- Return the value of text as a number, or throw an error.
        -- This supports extremely basic expressions of the form:
        --   a / b
        --   a ^ b
        -- where a and b are numbers or 'pi'.
        local n = tonumber(text)
        if n then
                return n
        end
        n = text:gsub('pi', math.pi)
        for _, op in ipairs({ '/', '^' }) do
                local a, b = n:match('^(.-)' .. op .. '(.*)$')
                if a then
                        a = tonumber(a)
                        b = tonumber(b)
                        if a and b then
                                if op == '/' then
                                        return a / b
                                elseif op == '^' then
                                        return a ^ b
                                end
                        end
                        break
                end
        end
        error('Unit "' .. ucode .. '" has invalid scale "' .. text .. '"')
end

local function get_builtin_unit(ucode, definitions)
```

```
	-- Return table of information for the specified built-in unit, or nil if
	-- Each defined unit code must be followed by two spaces (not tab charact
	local _, pos = definitions:find('\n' .. ucode .. '  ', 1, true)
	if pos then
		local endline = definitions:find('%s*\n', pos)
		if endline then
			local result = {}
			local n = 0
			local text = definitions:sub(pos + 1, endline - 1):gsub('
			for item in (text .. '\t'):gmatch('(%S.-)\t') do
				if item == 'ALIAS' then
					result.alias = true
				elseif item == 'ANGLE' then
					result.isangle = true
					result.nospace = true
				elseif item == 'NOSPACE' then
					result.nospace = true
				elseif item == 'SI' then
					result.si = true
				else
					n = n + 1
					if n == 1 then
						local link, symbol = item:match('
						if link then
							result.symbol = symbol
							result.link = link
							n = 2
						else
							result.symbol = item
						end
					elseif n == 2 then
						result.link = item
					elseif n == 3 then
						result.scale_text = item
						result.scale = get_scale(item, uc
					else
						result.more_ignored = item
						break
					end
				end
			end
			if result.si then
				local s = result.symbol
				if ucode == 'mc' .. s or ucode == 'mu' .. s then
					result.ucode = 'µ' .. s  -- unit code for
				end
			end
			if n >= 2 or (n >= 1 and result.alias) then
				return result
			end
			-- Ignore invalid definition, treating it as a comment.
		end
	end
end

local function convert_lookup(ucode, value, scaled_top, want_link, si, options)
	local lookup = require(convert_module)._unit
	return lookup(ucode, {
			value = value,
			scaled_top = scaled_top,
			link = want_link,
			si = si,
			sort = options.sortable,
		})
```

```
end

local function get_unit(ucode, value, scaled_top, options)
	local want_link = options.want_link
	if scaled_top then
		want_link = options.want_per_link
	end
	local data = mw.loadData(data_module)
	local result = options.want_longscale and
		get_builtin_unit(ucode, data.builtin_units_long_scale) or
		get_builtin_unit(ucode, data.builtin_units)
	local si, use_convert
	if result then
		if result.alias then
			ucode = result.symbol
			use_convert = true
		end
		if result.scale then
			-- Setting si means convert will use the unit as given, a
			-- will be calculated from the value without any extra sc
			-- occur if convert found the unit code. For example, if
			-- unit 'year' with a scale and if si were not set, conve
			-- its own scale because convert knows that a year is 31,
			si = { result.symbol, result.link }
			value = value * result.scale
		end
		if result.si then
			ucode = result.ucode or ucode
			si = { result.symbol, result.link }
			use_convert = true
		end
	else
		result = {}
		use_convert = true
	end
	local convert_unit = convert_lookup(ucode, value, scaled_top, want_link,
	result.sortkey = convert_unit.sortspan
	if use_convert then
		result.text = convert_unit.text
		result.scaled_top = convert_unit.scaled_value
	else
		if want_link then
			result.text = '[[' .. result.link .. '|' .. result.symbol
		else
			result.text = result.symbol
		end
		result.scaled_top = value
	end
	return result
end

local function makeunit(value, options)
	-- Return table of information for the requested unit and options, or
	-- return nil if no unit.
	options = options or {}
	local unit
	local ucode = options.u
	local percode = options.per
	if ucode then
		unit = get_unit(ucode, value, nil, options)
	elseif percode then
		unit = { nospace = true, scaled_top = value }
	else
		return nil
```

```
	end
	local text = unit.text or ''
	local sortkey = unit.sortkey
	if percode then
		local function bracketed(code, text)
			return code:find('[*./]') and '(' .. text .. ')' or text
		end
		local perunit = get_unit(percode, 1, unit.scaled_top, options)
		text = (ucode and bracketed(ucode, text) or '') ..
				'/' .. bracketed(percode, perunit.text)
		sortkey = perunit.sortkey
	end
	if not (unit.nospace or options.nospace) then
		text = ' ' .. text
	end
	return { text = text, isangle = unit.isangle, sortkey = sortkey }
end

local function list_units(mode)
	-- Return wikitext to list the built-in units.
	-- A unit code should not contain wikimarkup so don't bother escaping.
	local data = mw.loadData(data_module)
	local definitions = data.builtin_units .. data.builtin_units_long_scale
	local last_was_blank = true
	local n = 0
	local result = {}
	local function add(line)
		if line == '' then
			last_was_blank = true
		else
			if last_was_blank and n > 0 then
				n = n + 1
				result[n] = ''
			end
			last_was_blank = false
			n = n + 1
			result[n] = line
		end
	end
	local si_prefixes = {
		-- These are the prefixes recognized by convert; u is accepted fo
		y = 'y',
		z = 'z',
		a = 'a',
		f = 'f',
		p = 'p',
		n = 'n',
		u = 'µ',
		['µ'] = 'µ',
		m = 'm',
		c = 'c',
		d = 'd',
		da = 'da',
		h = 'h',
		k = 'k',
		M = 'M',
		G = 'G',
		T = 'T',
		P = 'P',
		E = 'E',
		Z = 'Z',
		Y = 'Y',
	}
	local function is_valid(ucode, unit)
```

```
                if unit and not unit.more_ignored then
                        assert(type(unit.symbol) == 'string' and unit.symbol ~= '
                        if unit.alias then
                                if unit.link or unit.scale_text or unit.si then
                                        return false
                                end
                        end
                        if unit.si then
                                if unit.scale_text then
                                        return false
                                end
                                ucode = unit.ucode or ucode
                                local base = unit.symbol
                                if ucode == base then
                                        unit.display = base
                                        return true
                                end
                                local plen = #ucode - #base
                                if plen > 0 then
                                        local prefix = si_prefixes[ucode:sub(1, p
                                        if prefix and ucode:sub(plen + 1) == base
                                                unit.display = prefix .. base
                                                return true
                                        end
                                end
                        else
                                unit.display = unit.symbol
                                return true
                        end
                end
                return false
        end
        local lookup = require(convert_module)._unit
        local function show_convert(ucode, unit)
                -- If a built-in unit defines a scale or sets the SI flag, any ur
                -- convert is not used (the scale or SI prefix's scale is used fo
                -- If there is no scale or SI flag, and the unit is not defined i
                -- the sort key may not be correct; this allows such units to be
                if not (unit.si or unit.scale_text) then
                        if mode == 'convert' then
                                unit.show = not lookup(unit.alias and unit.symbol
                                unit.show_text = 'CONVERT'
                        elseif mode == 'unknown' then
                                unit.show = lookup(unit.alias and unit.symbol or
                                unit.show_text = 'UNKNOWN'
                        elseif not unit.alias then
                                -- Show convert's scale in square brackets ('[1]
                                -- Don't show scale for an alias because it's mis
                                -- and an alias is probably not useful for anythi
                                local scale = lookup(ucode, {value=1, sort='on'})
                                if type(scale) == 'number' then
                                        scale = string.format('%.5g', scale):gsub
                                else
                                        scale = '?'
                                end
                                unit.show = true
                                unit.show_text = '[' .. scale .. ']'
                        end
                end
        end
        for line in definitions:gmatch('([^\n]*)\n') do
                local pos, _ = line:find('  ', 1, true)
                if pos then
                        local ucode = line:sub(1, pos - 1)
```

```
                        local unit = get_builtin_unit(ucode, '\n' .. line .. '\n
                        if is_valid(ucode, unit) then
                                show_convert(ucode, unit)
                                local flags, text
                                if unit.alias then
                                        text = unit.symbol
                                else
                                        text = '[[' .. unit.link .. '|' .. unit.c
                                end
                                if unit.isangle then
                                        unit.nospace = nil  -- don't show redunda
                                end
                                for _, f in ipairs({
                                                { 'alias', 'ALIAS' },
                                                { 'isangle', 'ANGLE' },
                                                { 'nospace', 'NOSPACE' },
                                                { 'si', 'SI' },
                                                { 'scale_text', unit.scale_text }
                                                { 'show', unit.show_text },
                                        }) do
                                        if unit[f[1]] then
                                                local t = f[2]
                                                if t:match('^%u+$') then
                                                        t = '<small>' .. t .. '</
                                                end
                                                if flags then
                                                        flags = flags .. ' ' .. t
                                                else
                                                        flags = t
                                                end
                                        end
                                end
                                if flags then
                                        text = text .. ' • ' .. flags
                                end
                                add(ucode .. ' = ' .. text .. '<br />')
                        else
                                add(line .. ' ◆ <b>invalid definition</b><br />')
                        end
                else
                        add(line)
                end
        end
        return table.concat(result, '\n')
end

local delimit_groups = require('Module:Gapnum').groups
local function delimit(sign, numstr, fmt)
        -- Return sign and numstr (unsigned digits or '.' only) after formatting
        -- Four-digit integers are not formatted with gaps.
        fmt = (fmt or ''):lower()
        if fmt == 'none' or (fmt == '' and #numstr == 4 and numstr:match('^%d+$')
                return sign .. numstr
        end
        -- Group number by integer and decimal parts.
        -- If there is no decimal part, delimit_groups returns only one table.
        local ipart, dpart = delimit_groups(numstr)
        local result
        if fmt == 'commas' then
                result = sign .. table.concat(ipart, ',')
                if dpart then
                        result = result .. '.' .. table.concat(dpart)
                end
        else
```

```
                -- Delimit with a small gap by default.
                local groups = {}
                groups[1] = table.remove(ipart, 1)
                for _, v in ipairs(ipart) do
                        table.insert(groups, '<span style="margin-left:.25em;">'
                end
                if dpart then
                        table.insert(groups, '.' .. (table.remove(dpart, 1) or '
                        for _, v in ipairs(dpart) do
                                table.insert(groups, '<span style="margin-left:.2
                        end
                end
                result = sign .. table.concat(groups)
        end
        return result
end

local function sup_sub(sup, sub, align)
        -- Return the same result as Module:Su except val defaults to align=right
        if align == 'l' or align == 'left' then
                align = 'left'
        elseif align == 'c' or align == 'center' then
                align = 'center'
        else
                align = 'right'
        end
        return '<span style="display:inline-block;margin-bottom:-0.3em;vertical-a
                align .. ';">' .. sup .. '<br />' .. sub .. '</span>'
end

local function range_text(items, unit_table, options)
        local fmt = options.fmt
        local nend = items.nend or ''
        if items.isrepeat or unit_table.isangle then
                nend = nend .. unit_table.text
        end
        local text = ''
        for i = 1, #items do
                if i % 2 == 0 then
                        text = text .. items[i]
                else
                        text = text .. delimit(items[i].sign, items[i].clean, fmt
                end
        end
        return text
end

local function uncertainty_text(uncertainty, unit_table, options)
        local angle, text, need_parens
        if unit_table.isangle then
                angle = unit_table.text
        end
        local upper = uncertainty.upper or {}
        local lower = uncertainty.lower or {}
        local uncU = upper.clean
        if uncU then
                local fmt = options.fmt
                local uncL = lower.clean
                if uncL then
                        uncU = delimit('+', uncU, fmt) .. (upper.errend or '')
                        uncL = delimit('−', uncL, fmt) .. (lower.errend or '')
                        if angle then
                                uncU = uncU .. angle
                                uncL = uncL .. angle
```

```
                        end
                        text = (angle or '') ..
                                '<span style="margin-left:0.3em;">' ..
                                sup_sub(uncU, uncL, options.align) ..
                                '</span>'
                else
                        if upper.parens then
                                text = '(' .. uncU .. ')'  -- old template did no
                        else
                                text = (angle or '') ..
                                        '<span style="margin-left:0.3em;margin-ri
                                        delimit('', uncU, fmt)
                                need_parens = true
                        end
                        if uncertainty.errend then
                                text = text .. uncertainty.errend
                        end
                        if angle then
                                text = text .. angle
                        end
                end
        else
                if angle then
                        text = angle
                end
        end
        return text, need_parens
end

local function _main(values, unit_spec, options)
        if options.sandbox then
                data_module = data_module .. '/sandbox'
                convert_module = convert_module .. '/sandbox'
        end
        local action = options.action
        if action then
                if action == 'list' then
                        -- Kludge: am using the align parameter (a=xxx) for type
                        return list_units(options.align)
                end
                return valerror('invalid action "' .. action .. '".', options.noc
        end
        local number = values.number or (values.numbers and values.numbers[1]) or
        local e_10 = options.e or {}
        local novalue = (number.value == nil and e_10.clean == nil)
        local fmt = options.fmt
        local want_sort = true
        local sortable = options.sortable
        if sortable == 'off' or (sortable == nil and novalue) then
                want_sort = false
        elseif sortable == 'debug' then
                -- Same as sortable = 'on' but the sort key is displayed.
        else
                sortable = 'on'
        end
        local sort_value = 1
        if want_sort then
                sort_value = number.value or 1
                if e_10.value and sort_value ~= 0 then
                        -- The 'if' avoids {{val|0|e=1234}} giving an invalid sor
                        sort_value = sort_value * 10^e_10.value
                end
        end
        local unit_table = makeunit(sort_value, {
```

```
                                                u = unit_spec.u,
                                                want_link = unit_spec.want_link,
                                                per = unit_spec.per,
                                                want_per_link = unit_spec.want_pe
                                                nospace = novalue,
                                                want_longscale = unit_spec.want_l
                                                sortable = sortable,
                                        })
        local sortkey
        if unit_table then
                if want_sort then
                        sortkey = unit_table.sortkey
                end
        else
                unit_table = { text = '' }
                if want_sort then
                        sortkey = convert_lookup('dummy', sort_value, nil, nil, r
                end
        end
        local final_unit = unit_table.isangle and '' or unit_table.text
        local e_text, n_text, need_parens
        local uncertainty = values.uncertainty
        if uncertainty then
                if number.clean then
                        n_text = delimit(number.sign, number.clean, fmt) .. (numb
                        local text
                        text, need_parens = uncertainty_text(uncertainty, unit_ta
                        if text then
                                n_text = n_text .. text
                        end
                else
                        n_text = ''
                end
        else
                if values.numbers.isrepeat then
                        final_unit = ''
                end
                n_text = range_text(values.numbers, unit_table, options)
                need_parens = true
        end
        if e_10.clean then
                if need_parens then
                        n_text = '(' .. n_text .. ')'
                end
                e_text = '10<sup>' .. delimit(e_10.sign, e_10.clean, fmt) .. '</s
                if number.clean then
                        e_text = '<span style="margin-left:0.25em;margin-right:0.
                end
        else
                e_text = ''
        end
        local result =
                (sortkey or '') ..
                (options.prefix or '') ..
                n_text ..
                e_text ..
                final_unit ..
                (options.suffix or '')
        if result ~= '' then
                result = '<span class="nowrap">' .. result .. '</span>'
        end
        return result .. (options.warning or '')
end
```

```
local function check_parameters(args, has_ranges, nocat)
	-- Return warning text for the first problem parameter found, or nothing
	local whitelist = {
		a = true,
		action = true,
		debug = true,
		e = true,
		['end'] = true,
		errend = true,
		['+errend'] = true,
		['-errend'] = true,
		fmt = true,
		['long scale'] = true,
		long_scale = true,
		longscale = true,
		nocategory = true,
		p = true,
		s = true,
		sortable = true,
		u = true,
		ul = true,
		up = true,
		upl = true,
	}
	for k, v in pairs(args) do
		if type(k) == 'string' and not whitelist[k] then
			local warning = 'Val parameter "' .. k .. '=' .. v .. '"
			return valerror(warning, nocat, true)
		end
	end
	if not has_ranges and args[4] then
		return valerror('Val parameter 4 ignored', nocat, true)
	end
end

local function main(frame)
	local getArgs = require('Module:Arguments').getArgs
	local args = getArgs(frame, {wrappers = { 'Template:Val' }})
	local nocat = args.nocategory
	local numbers = {}  -- table of number tables, perhaps with range text
	local msg = get_args(numbers, args)
	if msg then
		return valerror(msg, nocat)
	end
	if args.u and args.ul then
		return valerror('unit (<b>u</b>) and unit with link (<b>ul</b>) a
	end
	if args.up and args.upl then
		return valerror('unit per (<b>up</b>) and unit per with link (<b>
	end
	local values
	if numbers.has_ranges then
		-- Multiple values with range separators but no uncertainty.
		numbers.nend = args['end']
		values = {
			numbers = numbers,
		}
	else
		-- A single value with optional uncertainty.
		local function setfield(i, dst, src)
			local v = args[src]
			if v then
				if numbers[i] then
					numbers[i][dst] = v
```

```
                                else
                                        numbers[i] = { [dst] = v }
                                end
                        end
                end
                setfield(1, 'nend', 'end')
                setfield(2, 'errend', '+errend')
                setfield(3, 'errend', '-errend')
                values = {
                        number = numbers[1],
                        uncertainty = {
                                upper = numbers[2],
                                lower = numbers[3],
                                errend = args.errend,
                        }
                }
        end
        local unit_spec = {
                        u = args.ul or args.u,
                        want_link = args.ul ~= nil,
                        per = args.upl or args.up,
                        want_per_link = args.upl ~= nil,
                        want_longscale = (args.longscale or args.long_scale or ar
                }
        local options = {
                        action = args.action,
                        align = args.a,
                        e = numbers.e,
                        fmt = args.fmt,
                        nocat = nocat,
                        prefix = args.p,
                        sandbox = string.find(frame:getTitle(), 'sandbox', 1, tru
                        sortable = args.sortable or (args.debug == 'yes' and 'deb
                        suffix = args.s,
                        warning = check_parameters(args, numbers.has_ranges, noca
                }
        return _main(values, unit_spec, options)
end

return { main = main, _main = _main }
```

Modul:Val/units

The list of Val units is published at Template:Val/list, and here is the place that produces that report. So preview Vorlage:Tl from the edit box to see your changes before saving them. The file format and syntax are mostly self explanatory.

- The field separator is two or more spaces.
- You can enter new units in the "Unsorted units" section if you are not sure where else it might go.
- If the same *unit code* is defined twice on this page, the first one overrides the later one.
- For new entries the style guideline is Wikipedia:UNITS.
- Convert and Val share units. If you have an issue with a unit pagename or a unit symbol, and that unit is not published at Val/list, you may decide to address it at Template talk:Convert. To override entries at Convert, make an entry here.
- If you're not in a hurry, you may notice when editing Val/units that it consists of two Lua string assignments, and Lua comments. Be careful.

Questions or requests related to Val units are welcomed at Template talk:Val. For feedback specifically about the terminology or procedural steps seen on this page, please use the talk page.

Below are the detailed procedures, examples, descriptions of testing and previewing, explanations about sorting Val expressions, and links to helpful pages. There's also further information about Val/Convert relations.

Inhaltsverzeichnis

1 Introduction 17
2 How to add a unit 18
2.1 Examples 18
3 Testing a new unit 19
4 Advanced unit-entry format 20
4.1 Fourth field flags 21
4.2 Sorting 21
4.2.1 Scale 22
4.2.2 SI flag 22
5 Alias a Convert unit 23
6 File format 24
7 Advanced unit flags 24
8 Interaction with Convert 24
9 Notes 25
10 See also 25

Introduction

An entry defining a unit for Val is a single line under a section heading.[1] It starts with the **unit code**, followed by at least two spaces and a link. If you are adding a group of related units, you can enter a blank line around them to group them in the report at Val/list. An entry is ignored if it lacks at least *two adjacent space characters*.

*code*Vorlage:Spaces[[*pagename* | *symbol*]]

unit code

The keyboard typeable name of the symbol that users give Val's Vorlage:Para, Vorlage:Para, Vorlage:Para or Vorlage:Para parameters.

- Unit codes will be case sensitive.
- Prefer Vorlage:Code for the Greek letter µ, if you're not sure.
- Composite units have dimensions that multiply, divide, and apply powers to component units. Use Vorlage:Code to multiply, Vorlage:Code to divide, and a signed digit for powers. For multiplication, Vorlage:Code is deprecated. When adding a unit that includes division, consider also adding a version with multiplication by the negated power; for example, Vorlage:Code and Vorlage:Code (for m/s^2 and ms^2).

unit pagename

Title or section of an article. When linked with ul or upl, the title or one of its redirects can expand the abbreviation for the unit.[2]

unit symbol

Verifiable, standard symbol, formatted in accordance with WP:UNITS.

- Templates will not work for producing the unit symbol for input to this module; only Wikipedia HTML formatting is accepted.
- Composite units use Vorlage:Code to multiply, Vorlage:Code to divide, and Vorlage:Tag for powers. Division is also the unit-inversion form that multiplies a negative numbered power, for example for m/s^2 and ms^2 (from Vorlage:Code and Vorlage:Code).
- If the hover-text just shows the abbreviation, it is not a user-friendly unit-symbol. When not linked, the unit's abbreviation can be spelled out with hover-text at the symbol by way of the *title* attribute of either Vorlage:Tag or Vorlage:Tag.
- For more about HTML tags and HTML symbols such as Vorlage:Code, see Wikipedia:HTML.
- For more information about marking up your unit symbol see WP:HTML#formatting.
- Examples of unit codes, and hover text are at Template:Val/list.

unit code alias

The same unit/pagename/symbol, but by way of a different unit code.
Unit code aliases are commonly applied for

- capitalization, to make the unit code case insensitive.
- per units, for example m/s and msVorlage:Sup.
- Greek letters, to allow for both US keyboard and Greek-character-input applications, for example the SI prefix µ

Preview page with this template/module

A feature, similar to a sandbox and testcases, that provides a preview of how the code currently in the edit box (sandbox) will look when applied to any page (testcases).

The preview of main interest is `Template:Val/list`[3], but previewing other your own page of interest with your newly added unit in a val call and in a sorting table are also part of this procedure. Template:Val/units/testcases contains Val calls for all defined Val units.

How to add a unit

To maintain Val units,

1. Vorlage:Edit and make your changes.
2. Preview Vorlage:Space `Vorlage:Big` Vorlage:Space .
   - Look for any messages. An "Invalid definition" message is available automatically.
   - Prove the intended link from the preview.
   - Hover the mouse over the link, and read the hover-text or URL display somewhere in the browser.
3. Add any unit-code aliases.
4. Add any sorting if needed. Details about sorting are covered below.
5. Preview a test page. It will have template Val calls on it, and it may have a sortable table to test sorting. Template:Val/units/testcases is a test page containing Val calls for all units.
6. *Show changes* to prove no accidental edits occurred.
7. Save the page. Saving the page activates the changes immediately, and they go live. You're done adding your unit.

If you want a unit to add for practice, add one from List of common physics notations, or from SI units#Units and prefixes.

Changing or removing existing unit codes is possible by employing Vorlage:Tl to see how Val unit codes may or may not be in use on the wiki. For example, to see about changing or removing unit code `J.s`, do a

`{{tlusage|val|"J.s"|0}}` → Vorlage:Tlusage.

Put any unit code in quotes if it contains dash, dot, or slashVorlage:Mdashanything but a letter or number. See Template:Val/units/test for a list of these searches for each Val unit.

Examples

Say you're creating a new page or revamping an old page, and discover the need for a convenient way to make several entries containing c_0, and link that symbol to the page *Speed of light*. The following entry will define your unit code as `c0`, your unit symbol as `''c''<sub>0</sub>` , and the unit's article as *Speed of light#Numerical value, notation, and units*.

```
c0Vorlage:Spaces|Vorlage:Big|Speed of light#Numerical value, notation, and unit symbol|Vorlage:Big|''c''<sub>0</sub>|Vorlage:Big
```

or as explained below at §Advanced unit entry formats, you can also write

```
c0Vorlage:Spaces''c''<sub>0</sub>Vorlage:SpacesSpeed of light#Numerical value, notation, and units
```

Then preview with *Template:Val/list*, and check for an error message next to the new unit, and test the link you gave.

After that the page with the (saved) Val calls is used to test the linked and non-linked versions of the normal and the per units:

- `{{val|0.891|u=c0}}` → Vorlage:Val
- `{{val|0.891|ul=c0}}` → Vorlage:Val
- `{{val|0.891|up=c0}}` → Vorlage:Val
- `{{val|0.891|upl=c0}}` → Vorlage:Val

For an entire example that uses the other format to make a Val/unit entry.

1. Put this in a sandbox: `{{val|1.23|ul=`**`tins`**`}}` → Vorlage:Val
2. Edit Module:Val/units and insert a line like the following (do not save yet):

   ```
   tinsVorlage:Spacestins Vorlage:SpacesContins unities
   ```
3. Under "Preview page with this module" enter `Template:Val/list`, and click *Show preview*. It shows Val/list through the version of Val/units in the edit box. Say there are no errors, the markup and hover-text look good, and the link navigates to the unit's page.
4. Then in the same way of previewing, put the fullpagename of the sandbox from step 1, and click *Show preview*.
5. *Save page* to save the edit to Module:Val/units.

If you want a unit to add for practice, add one from *List of common physics notations*, or from *SI units#Units and prefixes*. There are many examples of *composite* units that have their own page, so adding a unit code for one of those should link to its page. There are many articles that could use a new Val unit, such as molarity at Resveratrol.

Either of the test pages of these examples could have the sortable test-table shown in the next section.

Testing a new unit

To test a *newly added* unit not used on any page, you will need to run the preview on a sandbox page you have already created. Here are all the test cases you can preview there before saving your changes here; they are the four Vorlage:Para parameters:

```
{{Val|9|u =            }}
{{Val|9|ul =           }}
{{Val|9|u=foo|up =     }}
{{Val|9|u=foo|upl =    }}
```

and the sortable table:

```
{| class="wikitable sortable" summary="Sortable table to test Val sorting"
! Val number and unit
|-
| {{val|5|u=      }}
|-
| {{val|3|u=      }}
|-
| {{val|1|u=      }}
|-
| {{val|2|u=      }}
|-
| {{val|4|u=      }}
|}
```

This table falls into place unsorted, so when your unit accepts an SI prefix you can test, say, k, m, and G, with your unit, and compare with e notations 1e3, 1e6, and 1e9 in the number. For example, Val sorts these two as equal: `1e3 m` (standard e notation) and `1 km`.

What to look for

- The linked and non-linked markup should look exactly the same.
- Navigate to the new link. It is safe: you can go back in your browser to here.
- The two Vorlage:Para versions should have no space in front of them.
- For SI prefixes sorting 2e3 (or 2000) should be greater than k (kilo prefix).

Advanced unit-entry format

In the usual format

code`Vorlage:Spaces[[`*pagename*`|`*symbol*`]]`

the wikilink represents two fields itself, for a total of three fields per entry. But you cannot use a wikilink for composite units that need more than one pagename.

The other record type for adding a unit entry is also three fields. It has the same three fields, but they are in a different order.

code`Vorlage:Spaces`*symbol*`Vorlage:Spaces`*pagename*

This format separates each field with whitespace, and also takes tabs between fields two and three.

For a new composite unit you should probably link the whole composition, or link the largest portion which could have its own page.

- The val user can compose a *divisor* unit on the fly from existing unit codes, and with individually linked numerator and denominator. For example:

  `{{val|99|ul=m|upl=d}}` → Vorlage:Val

- The val user can compose a *multiplier* unit on the fly by using the Vorlage:Para parameter to prepend to the unit, and *these* can also be individually linked. For example (in geology) there is already Vorlage:Val to use with Vorlage:Para:

  `{{val|333|ul=uBP|end= [[megaannum|Ma]]}}` → Vorlage:Val".

For example Template:Val/list says

`m.s-1  Vorlage:BigMetre per second|m⋅s<sup>−1</sup>Vorlage:Big` linking to an article titled *Metre per second*, not

`m.s-1  Vorlage:BigMetre|mVorlage:Big⋅Vorlage:BigSecond|sVorlage:Big<sup>−1</sup>` which has separate links to already existing unit codes.

Fourth field flags

The module must be told directly about sorting factors, spacing, and aliasing for a unit code.

There is an optional field that goes at the end after two or more spaces or one or more tabs. It is a flag mainly used to provide for that unit to be sorted in a table. Flags are mainly for sorting, and they work for either record type. Just add two or more spaces, or one or more tabs, and then the flag field. (Optional flags ALIAS and NOSPACE and ANGLE are for even more advanced users. See §Advanced unit flags below.)

Using SI requires that the unit symbol compare precisely to the unit code, and so never allows HTML or other characters in the symbol. Any difference between the unit symbol and unit code must be an SI prefix, such as k, M, or G.

Sorting

Vorlage:Details Vorlage:Details

Val's sorting scale factor is for comparison to other Val units that might be sorted with it.

Where Sorting is done on the wiki, it is done in sortable tables. Val entries in a sortable table will need a fourth field sorting flag. It can be a number, an equation, or an `SI`, but it flags the same function: a wikitable sorting "scale".

To display the sort key use Vorlage:Para. For example

- `{{val|999|u=uV|debug=yes}}` → 6996998999999999999♠999 µV
- `{{val|99|u=V|debug=yes}}` → 7001990000000000000♠99 V

- {{val|1|u=kV|debug=yes}} → 7003100000000000000♠1 kV

Scale

For scaling a unit to sort properly, you need to pick a number for a sorting factor. There are numerous examples at Template:Val/list and at Template:Val/sortkey/unit. A system of units will have its base units, for example 1 bit; then the scale for sorting a kilobyte unit is then 8000 (eight bits per byte, times a kilo, or thousand). Or a year scale is seconds so that all *times* sort by seconds, which is a base unit. In general the scale shows to be "base unit" of the same type times the "SI prefix", and if it's not that simple, then the unit system's number has associated a number to it, such as Avogadro's number.

For example, the following defines a unit with code `billion`, symbol `billion`, link 1,000,000,000 , and scale `1e9` (Vorlage:Val). After the following entry is saved to the database

```
billion             billion          1,000,000,000          1e9
```

{{val|2|u=billion}} would start sorting after {{val|98.7|e=3}}.

SI flag

Vorlage:Common metric prefixes SI is used because it scales Val expressions automatically, and it is a clean indicator that the unit will sort properly. It correctly scales any SI prefix for sorting, but not other unit codes.

For SI the unit symbol will not accept HTML, but will accept µ. HTML is not accepted at this time because in order to validate the entry, the unit code must differ from the unit symbol by exactly one valid SI prefix. If there is no difference, or too much difference, it is an invalid definition for sorting purposes. The exception is for the Greek letter µ: if you used a character input application to "install" the Greek letter µ in your symbol, for your "easy to type" unit code, Vorlage:Key, these two are not a character mismatch.

All unit entries that use SI will have the same base unit as the symbol at Val/units, but they will display properly at Val/list.

For example, kilo is a thousand, but you're defining km2 for kilometers squared, and need HTML. You can't use SI with HTML, so use 1000*1000, or 1000000 in the sorting field. Use 1e-6 or 0.000001 or 1/100000 instead.

If the unit you are maintaining has SI prefixes and they are all likely to be sorted in a table, add up to twelve entries, one for each common SI unit. Some of these, like *Meter* in the example, may have their own article, but usually all go to the base unit's pagename. Here is how *meter* is defined.

```
m     [[Metre|m]]          SI
cm    [[Centimetre|m]]     SI
dam   [[Decametre|m]]      SI
dm    [[Decimetre|m]]      SI
hm    [[Hectometre|m]]     SI
```

```
km    [[Kilometre|m]]     SI
Mm    [[Megametre|m]]     SI
mm    [[Millimetre|m]]    SI
um    [[Micrometre|µm]]   1/1000000
µm    [[Micrometre|m]]    SI
nm    [[Nanometre|m]]     SI
pm    [[Picometre|m]]     SI
```

The information that was in the unit symbol is now fully specified in, and exhibited at, the unit code. SI specifies that the unit's symbol has been transformed from a symbol to a string for use in string comparison that will finally result in calculating a sorting factor. It conveniently uses the idea that the unit code is often equal to the symbol, especially with SI units. The field definitions are sacrificed for a simplicity in the user presentation, user calculation, and user entry.

In the other format, the following defines three unit codes for volts, V for sorting. V is the base unit with the SI prefix removed. A unit code defined in this manner will have its sort key scaled by the software according to the SI prefix produced by the difference between the unit code and unit symbol.

```
kV   V   Kilovolt    SI
µV   V   Microvolt   SI
uV   V   Microvolt   SI
```

Now {{val|1|u=kV}} will sort after {{val|999|u=V}} without having to resort to using a number, and with the clean representation at Val/list.

The symbol column shows "V" for each, but it is not the symbol—it is the base unit after removing the SI prefix so convert can work out what is intended to be the prefix. The following would give identical results:

```
kV   kV   Kilovolt    1e3
µV   µV   Microvolt   1e-6
uV   µV   Microvolt   1e-6
```

As you can see, without "SI", you define both the symbol, and the scale. You define the symbol with HTML or the Greek letter or other symbol, and you define the scale with a number or an equation. When "SI" is used, convert just does the right thing for the symbol and scale.

Alias a Convert unit

If you are here to change the link or markup of a unit, but it is not listed at Val/units, sometimes you can find the unit markup and link that you do want, already existing at Template: Convert#Units. In that case you can change the unit code to whatever you'd prefer, and it will achieve your goal. For example, if {{Val|1|C}} is going to Celsius instead of Coulombs, you can define your own unit code, say "degC".

The following defines **degC** to refer to the unit known as **°C** in convert. There is no link because a link is defined at Convert.

```
degC              °C              ALIAS
```

File format

If you want to reorganize sections here, note that the two lines `local builtin_units` and `local builtin_units_long_scale` require a blank line after them. The section long scale, with all the units like "billions" and "trillions", is under the latter, near the bottom of the page. All the rest of the units are in the former.

The file format is two Lua strings and a return statement with them in it: a string in quotes [=[... builtin_units ...]=], and another string in quotes [=[... builtin_units_long_scale ...]=]. The first string, builtin_units, is short-scale, second string is long scale. The reason there are two strings is because of the difference between British and US terms surrounding "billion", "billionth", etc.

For each string there must be a blank line before the first line of the string and after the last line of the string. In other words the first two and last two characters of each string must be newlines.

There is one record per line, starting in first column, having 2-4 fields. The field separator is two or more spaces. Between first and second fields, use two or more spaces. Between all other fields, use two or more spaces, or one or more tabs. Entries without two spaces in them are ignored.

Advanced unit flags

You can alias Convert or Val units. But these are different things.

- A "unit code alias" is when the same *unit pagename and unit symbol* are defined twice. If a different *unit code* is assigned to the same unit symbol and unit pagename, (say, as a copy of the previous entry) it will work as a Val alias.
- A "unit alias" is when Val alias a Convert unit code. A unit of measurement is *here* denoted ALIAS to mean "they are defined *there*". Val *defaults* to Convert, but it's good to this explicit for certain Val units: the ones tempting to define here, but that you don't want defined here because, says ALIAS, they are *already* defined there.
- If your unit code is not listed at Vorlage:Tl, you can check for it at Vorlage:Tl, or at Vorlage:Tl

For spacing and aliasing:

- `ALIAS` specifies that the unit's symbol is the code for a unit defined in Vorlage:Tlf}.
- `NOSPACE` prevents the default insertion of a non-breaking space before the unit symbol.
- If you see the `ANGLE` flag, this identifies those special units that must displayed not only after the main Val number, but also after its uncertainty#Measurements numbers.(`ANGLE` also implies `NOSPACE`.)

Interaction with Convert

Val and Convert share unit codes, but their units of measurement are different.

- Most of the wiki's unit codes are managed by Vorlage:Tl.

- Some very few unit codes here, like C and F, mean something different there. Val caters to Coulombs and Farads, while Convert caters to Celsius and Fahrenheit. Convert and Val *unit codes* are mostly identical, like they are for degC and for degF.
- Val could need *any* unit, while not all units are needed in conversions, so Val needs *all* of Convert's units and some of its own.
- Sorting functionality is handled for Val by Vorlage:Tlf.)
- At Convert the procedure for defining a unit is much more involved than it is here, because there every unit defined must reference associated units, conversion factors, alternate spellings, and many other attributes. An entry at Convert is defined as a multi-line, multi-attribute Lua table with its attendant syntax, and inside a larger Lua script. Therefore Convert is more conservative about adding units. Val may be more liberal in this respect if only because unit entry is simpler and "wikified". Therefore WP:Be bold. Bold customization may be a worthwhile risk at Val/units, or it may get removed. Although units used in *articles* have definite stylistic standards, there remains room for depending on Val for markup (but see WP:Accessibility about color, link, and text). You may want to customize some Val unit codes that will automate some Val markup for special articles, the *talk page*, etc. See WP:HTML#formatting for possibilities.

Notes

Vorlage:Refs

See also

- Module:Val
- Template:Val/units/test
- Template:Val/units/testcases
- Template:Val/units/loadtest
- Template:Val/unitsfromconvert

---

1. ↑ Module pages don't have section editing.
2. ↑ A redirect page is easy to make; there are tools on the editor toolbars.
3. ↑ The special relation between Val/units and Val/list is that the wikitext of Val/list is always generated by Val/units when that page is viewed, *so* when you **preview page with this module**, Val/list is generated using the contents of the edit box of Val/units.

```
-- Definitions for units known to val
-- File format is two strings and a return statement with them in it:
-- string in quotes [=[ ... builtin_units ... ]=].
-- string in quotes [=[ ... builtin_units_long_scale ... ]=].
-- First string, builtin_units, is short-scale, second string is long scale.

-- Entry format:
-- One record per line, starting in first column, having 2-4 fields.
-- Field separator: two or more spaces
-- Between first and second fields: two or more spaces
```

```
-- Between all other fields: two or more spaces, or one or more tabs
-- Entries without two spaces in them are ignored.

-- There must be a blank line before the first entry and after the last.
-- I.e. the first two and last two characters of the string must be newlines.

-- Format of entry. Two record types:
--
-- One record type is a wikilink:
-- Unit-code        [[ pagename | Symbol-accepts-HTML-only ]]
-- Text-field separator is still two spaces.  Two spaces not allowed in wikilink
--
-- The other record type is all fields:
-- Unit-code        symbol-accepts-HTML-only        pagename#section-OK
--
-- Plus there is an optional field that goes at the end after two or more spaces
-- Whether it is a number or an equation or the letters SI,
-- any of these three has the same function: a wikitable sorting "scale".
-- It is for sorting, and it works for either record type.
-- Difference is SI can't accept HTML. But SI correctly scales any SI prefix.
-- (Optional fields ALIAS and NOSPACE and ANGLE are for advanced users.)

-- "Invalid unit" error:
-- Using SI requires that the symbol equal unit-code, so never allows HTML.
-- Any difference between SI or symbol must be an SI prefix, such as k, M, or G.
-- A space at the end of an entry is an error. No space at each EOL.

local builtin_units = [=[

== Test ==
Foo  [[Hz|<samp>Foo</samp>]]
Baz  [[Hertz|baz<sub>0</sub>]]
Baz  [[Kelvins|baz<sub>0</sub>]]
Bar  [[Foobar|bar<abbr title="super duper">0</abbr>]]
quux  [[Foobar|<span title="super duper 2">bar0</span>]]

== Unsorted units ==
c0  [[Speed of light#Numerical value, notation, and units|''c''<sub>0</sub>]]
lbf  [[Pound (force)|<span title="pound-force">lb<sub>F</sub></span> ]]
N.s  [[Newton-second|N&sdot;s]]
J.K-1  [[Joule per kelvin|J&sdot;K<sup>−1</sup>]]
C.mol-1  [[Faraday constant|C&sdot;mol<sup>−1</sup>]]
C/mol  [[Faraday constant|C/mol]]
C.kg-1  [[Roentgen (unit)|C&sdot;kg<sup>−1</sup>]]
C/kg  [[Roentgen (unit)|C/kg]]
F.m-1  [[vacuum permittivity|F&sdot;m<sup>−1</sup>]]
F/m  [[vacuum permittivity|F/m]]
e  [[Elementary charge|''e'']]
kB  [[Kilobyte|kB]]  8e3
MB  [[Megabyte|MB]]  8e6
GB  [[Gigabyte|GB]]  8e9
TB  [[Terabyte|TB]]  8e12
lx  [[Lux (unit)|lx]]
nat  [[nat (unit)|nat]]

== Time and frequency ==
byte/s  [[Data rate units|byte/s]]  8
kB/s  [[Data rate units#Kilobyte per second|<span title="Kilobytes per second">kB
MB/s  [[Data rate units#Megabyte per second|<span title="Megabytes per second">MB
GB/s  [[Data rate units#Gigabyte per second|<span title="Gigabytes per second">GB
TB/s  [[Data rate units#Terabyte per second|<span title="Terabytes per second">TB
bit/s  [[Bit per second|bit/s]]  1
bps  [[Bit per second|bit/s]]  1
kbit/s  [[Kilobit per second|kbit/s]]  1e3
```

```
Mbit/s  [[Megabit per second|Mbit/s]]  1e6
Gbit/s  [[Gigabit per second|Gbit/s]]  1e9
Tbit/s  [[Terabit per second|Tbit/s]]  1e12
kT/s  [[Transfer (computing)|<span title="Kilotransfers per second">kT/s</span>]]
MT/s  [[Transfer (computing)|<span title="Megatransfers per second">MT/s</span>]]
GT/s  [[Transfer (computing)|<span title="Gigatransfers per second">GT/s</span>]]
year  [[Year|year]]  31557600
years  [[Year|years]]  31557600
yr  [[Year#Symbols y and yr|yr]]  31557600
y  [[Year|y]]  31557600
a  [[Annum|a]]  31557600
Ga  [[Gigaannum|Ga]]  31557600000000000
Ma  [[Megaannum|Ma]]  31557600000000
ka  [[Kiloannum|ka]]  31557600000
kyr  [[kyr|kyr]]  31557600000
kya  [[kyr|kya]]  31557600000
myr  [[myr|myr]]  31557600000000
mya  [[Mya (unit)|mya]]  31557600000000
byr  [[Billion years|byr]]  315576000000000000
bya  [[Billion years ago|bya]]  315576000000000000
Gyr  [[billion years|Gyr]]  315576000000000000
BP  [[Before present|BP]]
uBP  [[Radiocarbon dating#Calibration|<sup>14</sup>C yr BP]]
BC  [[Before Christ|BC]]  -1
AD  [[Anno Domini|AD]]  1
BCE  [[Before the Common Era|BCE]]  -1
CE  [[Common Era|CE]]  1
JD  [[Julian date|JD]]  1
MJD  [[Modified Julian date|MJD]]  1

s-1  [[Second|s<sup>−1</sup>]]
s-2  [[Second|s<sup>−2</sup>]]
s2  [[Second|s<sup>2</sup>]]

s  [[Second|s]]  SI
as  [[Attosecond|s]]  SI
cs  [[Second|s]]  SI
das  [[Second|s]]  SI
ds  [[Second|s]]  SI
Es  [[Second|s]]  SI
fs  [[Femtosecond|s]]  SI
Gs  [[Second|s]]  SI
hs  [[Second|s]]  SI
ks  [[Second|s]]  SI
ms  [[Millisecond|s]]  SI
µs  [[Microsecond|s]]  SI
us  [[Microsecond|s]]  SI
Ms  [[Second|s]]  SI
ns  [[Nanosecond|s]]  SI
ps  [[Picosecond|s]]  SI
Ps  [[Second|s]]  SI
Ts  [[Second|s]]  SI
Ys  [[Second|s]]  SI
ys  [[Yoctosecond|s]]  SI
Zs  [[Second|s]]  SI
zs  [[Zeptosecond|s]]  SI

Hz  [[Hertz|Hz]]  SI
aHz  [[Hertz|Hz]]  SI
cHz  [[Hertz|Hz]]  SI
daHz  [[Hertz|Hz]]  SI
dHz  [[Hertz|Hz]]  SI
EHz  [[Hertz|Hz]]  SI
fHz  [[Hertz|Hz]]  SI
```

```
hHz  [[Hertz|Hz]]  SI
GHz  [[Gigahertz|Hz]]  SI
kHz  [[Kilohertz|Hz]]  SI
MHz  [[Megahertz|Hz]]  SI
mHz  [[Hertz|Hz]]  SI
uHz  [[Hertz|Hz]]  SI
µHz  [[Hertz|Hz]]  SI
nHz  [[Hertz|Hz]]  SI
pHz  [[Hertz|Hz]]  SI
PHz  [[Hertz|Hz]]  SI
THz  [[Hertz|Hz]]  SI
yHz  [[Hertz|Hz]]  SI
YHz  [[Hertz|Hz]]  SI
zHz  [[Hertz|Hz]]  SI
ZHz  [[Hertz|Hz]]  SI

== Length, area, volume ==
Å3   [[Ångström|Å<sup>3</sup>]]
fb-1  [[Barn (unit)|fb<sup>−1</sup>]]
m-1  [[Metre|m<sup>−1</sup>]]
m-2  [[Square metre|m<sup>−2</sup>]]
m-3  [[Cubic metre|m<sup>−3</sup>]]
km2  [[Square kilometre|km<sup>2</sup>]]
km3  [[Cubic kilometre|km<sup>3</sup>]]
µm2  [[Square metre|µm<sup>2</sup>]]
um2  [[Square metre|µm<sup>2</sup>]]
am2  [[Square metre|am<sup>2</sup>]]
cm2  [[Square centimetre|cm<sup>2</sup>]]
dam2  [[Square metre|dam<sup>2</sup>]]
dm2  [[Square metre|dm<sup>2</sup>]]
Em2  [[Square metre|Em<sup>2</sup>]]
fm2  [[Square metre|fm<sup>2</sup>]]
Gm2  [[Square metre|Gm<sup>2</sup>]]
hm2  [[Square metre|hm<sup>2</sup>]]
mm2  [[Square metre|mm<sup>2</sup>]]
Mm2  [[Square metre|Mm<sup>2</sup>]]
nm2  [[Square metre|nm<sup>2</sup>]]
pm2  [[Square metre|pm<sup>2</sup>]]
Pm2  [[Square metre|Pm<sup>2</sup>]]
Tm2  [[Square metre|Tm<sup>2</sup>]]
ym2  [[Square metre|ym<sup>2</sup>]]
Ym2  [[Square metre|Ym<sup>2</sup>]]
zm2  [[Square metre|zm<sup>2</sup>]]
Zm2  [[Square metre|Zm<sup>2</sup>]]
gal  [[Gallon|gal]]
Gal  [[Gal (unit)|Gal]]
uGal  [[Gal (unit)|µGal]]
µGal  [[Gal (unit)|µGal]]
mGal  [[Gal (unit)|mGal]]

b  [[Barn (unit)|b]]  SI
ab  [[Barn (unit)|b]]  SI
cb  [[Barn (unit)|b]]  SI
dab  [[Barn (unit)|b]]  SI
db  [[Barn (unit)|b]]  SI
Eb  [[Barn (unit)|b]]  SI
fb  [[Barn (unit)|b]]  SI
Gb  [[Barn (unit)|b]]  SI
hb  [[Barn (unit)|b]]  SI
kb  [[Barn (unit)|b]]  SI
mb  [[Barn (unit)|b]]  SI
µb  [[Barn (unit)|b]]  SI
ub  [[Barn (unit)|b]]  SI
Mb  [[Barn (unit)|b]]  SI
```

```
nb  [[Barn (unit)|b]]  SI
pb  [[Barn (unit)|b]]  SI
Pb  [[Barn (unit)|b]]  SI
Tb  [[Barn (unit)|b]]  SI
Yb  [[Barn (unit)|b]]  SI
yb  [[Barn (unit)|b]]  SI
Zb  [[Barn (unit)|b]]  SI
zb  [[Barn (unit)|b]]  SI

== Velocity and acceleration ==
m.s-2  [[Metre per second squared|m&sdot;s<sup>−2</sup>]]
m/s2  [[Metre per second squared|m/s<sup>2</sup>]]
m.s-1  [[Metre per second|m&sdot;s<sup>−1</sup>]]
m/s  [[Metre per second|m/s]]
km.s-1  [[Metre per second|km&sdot;s<sup>−1</sup>]]
km/s  [[Metre per second|km/s]]

== Mass and energy ==
lbm  [[Pound (mass)|<span title="pound-mass">lb<sub>m</sub></span>]]
uJ  [[Joule|µJ]]
J.s  [[Joule-second|J&sdot;s]]
kWh  [[Kilowatt hour|kWh]]
kW.h  [[Kilowatt hour|kW&sdot;h]]
J/C  [[Volt|J/C]]
J/kg  [[Joule|J/kg]]

Da  [[Dalton (unit)|Da]]  SI
EDa  [[Dalton (unit)|Da]]  SI
PDa  [[Dalton (unit)|Da]]  SI
TDa  [[Dalton (unit)|Da]]  SI
GDa  [[Dalton (unit)|Da]]  SI
MDa  [[Dalton (unit)|Da]]  SI
kDa  [[Dalton (unit)|Da]]  SI
mDa  [[Dalton (unit)|Da]]  SI
uDa  [[Dalton (unit)|Da]]  SI
µDa  [[Dalton (unit)|Da]]  SI
nDa  [[Dalton (unit)|Da]]  SI
pDa  [[Dalton (unit)|Da]]  SI
fDa  [[Dalton (unit)|Da]]  SI
aDa  [[Dalton (unit)|Da]]  SI

g  [[Gram|g]]  SI
ag  [[Attogram|g]]  SI
cg  [[Centigram|g]]  SI
dag  [[Gram|g]]  SI
dg  [[Decigram|g]]  SI
Eg  [[Exagram|g]]  SI
fg  [[Femtogram|g]]  SI
Gg  [[Gigagram|g]]  SI
hg  [[Kilogram#SI multiples|g]]  SI
kg  [[Kilogram|g]]  SI
mcg  [[Microgram|g]]  SI
Mg  [[Megagram|g]]  SI
mg  [[Milligram|g]]  SI
ug  [[Microgram|g]]  SI
µg  [[Microgram|g]]  SI
ng  [[Nanogram|g]]  SI
Pg  [[Petagram|g]]  SI
pg  [[Picogram|g]]  SI
Tg  [[Tonne|g]]  SI
yg  [[Yoctogram|g]]  SI
Yg  [[Yottagram|g]]  SI
zg  [[Zeptogram|g]]  SI
Zg  [[Zettagram|g]]  SI
```

```
== Pressure and density ==
psi  [[Pounds per square inch|psi]]
g.cm-3  [[Gram per cubic centimetre|g&sdot;cm<sup>−3</sup>]]
g/cm3  [[Gram per cubic centimetre|g/cm<sup>3</sup>]]
kg.m-3  [[Kilogram per cubic metre|kg&sdot;m<sup>−3</sup>]]
kg/m3  [[Kilogram per cubic metre|kg/m<sup>3</sup>]]
kg/cm3  [[Density#Formula and common units|kg/cm<sup>3</sup>]]
g/L  [[Gram per litre|g/L]]
g/l  [[Gram per litre|g/l]]
mcg/dL  [[Gram per litre|µg/dL]]
mcg/dl  [[Gram per litre|µg/dl]]
mg/mL  [[Gram per litre|mg/mL]]
mg/ml  [[Gram per litre|mg/ml]]
ug/dL  [[Gram per litre|µg/dL]]
ug/dl  [[Gram per litre|µg/dl]]
µg/dL  [[Gram per litre|µg/dL]]
µg/dl  [[Gram per litre|µg/dl]]
mg.L-1  [[Gram per litre|<abbr title="milligrams per liter">mg/L</abbr>]]
mg/L  [[Gram per litre|<abbr title="milligrams per liter">mg/L</abbr>]]
mg.l-1  [[Gram per litre|<abbr title="milligrams per liter">mg/l</abbr>]]
mg/l  [[Gram per litre|<abbr title="milligrams per liter">mg/l</abbr>]]

== Fracture toughness ==
MPa.m.5  [[Fracture toughness|MPa&sdot;m<sup>1/2</sup>]]
kPa.m.5  [[Fracture toughness|kPa&sdot;m<sup>1/2</sup>]]
Pa.m.5  [[Fracture toughness|Pa&sdot;m<sup>1/2</sup>]]

== Temperature ==
degC  °C  ALIAS
degF  °F  ALIAS
degR  °R  ALIAS

K  [[Kelvin|K]]  SI
YK  [[Yottakelvin|K]]  SI
ZK  [[Zettakelvin|K]]  SI
EK  [[Kelvin|K]]  SI
PK  [[Petakelvin|K]]  SI
TK  [[Terakelvin|K]]  SI
GK  [[Gigakelvin|K]]  SI
MK  [[Megakelvin|K]]  SI
kK  [[Kilokelvin|K]]  SI
hK  [[Hectokelvin|K]]  SI
daK  [[Decakelvin|K]]  SI
dK  [[Decikelvin|K]]  SI
cK  [[Centikelvin|K]]  SI
mK  [[Millikelvin|K]]  SI
µK  [[Microkelvin|K]]  SI
uK  [[Microkelvin|K]]  SI
nK  [[Nanokelvin|K]]  SI
pK  [[Picokelvin|K]]  SI
fK  [[Femtokelvin|K]]  SI
aK  [[Attokelvin|K]]  SI
zK  [[Zeptokelvin|K]]  SI
yK  [[Yoctokelvin|K]]  SI

== Electromagnetism ==
Wb  [[Weber (unit)|Wb]]
N.A-2  [[Permeability (electromagnetism)|N&sdot;A<sup>−2</sup>]]
H.m-1  [[Permeability (electromagnetism)|H&sdot;m<sup>−1</sup>]]
V.m-1  [[Electric field|V&sdot;m<sup>−1</sup>]]
V/m  [[Electric field|V/m]]

C  [[Coulomb|C]]  SI
```

```
YC  [[Coulomb|C]]  SI
ZC  [[Coulomb|C]]  SI
EC  [[Coulomb|C]]  SI
PC  [[Coulomb|C]]  SI
TC  [[Coulomb|C]]  SI
GC  [[Coulomb|C]]  SI
MC  [[Coulomb|C]]  SI
kC  [[Coulomb|C]]  SI
hC  [[Coulomb|C]]  SI
daC  [[Coulomb|C]]  SI
dC  [[Coulomb|C]]  SI
cC  [[Coulomb|C]]  SI
mC  [[Coulomb|C]]  SI
µC  [[Coulomb|C]]  SI
uC  [[Coulomb|C]]  SI
nC  [[Coulomb|C]]  SI
pC  [[Coulomb|C]]  SI
fC  [[Coulomb|C]]  SI
aC  [[Coulomb|C]]  SI
zC  [[Coulomb|C]]  SI
yC  [[Coulomb|C]]  SI

F  [[Farad|F]]  SI
YF  [[Farad|F]]  SI
ZF  [[Farad|F]]  SI
EF  [[Farad|F]]  SI
PF  [[Farad|F]]  SI
TF  [[Farad|F]]  SI
GF  [[Farad|F]]  SI
MF  [[Farad|F]]  SI
kF  [[Farad|F]]  SI
hF  [[Farad|F]]  SI
daF  [[Farad|F]]  SI
dF  [[Farad|F]]  SI
cF  [[Farad|F]]  SI
mF  [[Farad|F]]  SI
µF  [[Farad|F]]  SI
uF  [[Farad|F]]  SI
nF  [[Farad|F]]  SI
pF  [[Farad|F]]  SI
fF  [[Farad|F]]  SI
aF  [[Farad|F]]  SI
zF  [[Farad|F]]  SI
yF  [[Farad|F]]  SI

H  [[Henry (unit)|H]]  SI
YH  [[Henry (unit)|H]]  SI
ZH  [[Henry (unit)|H]]  SI
EH  [[Henry (unit)|H]]  SI
PH  [[Henry (unit)|H]]  SI
TH  [[Henry (unit)|H]]  SI
GH  [[Henry (unit)|H]]  SI
MH  [[Henry (unit)|H]]  SI
kH  [[Henry (unit)|H]]  SI
hH  [[Henry (unit)|H]]  SI
daH  [[Henry (unit)|H]]  SI
dH  [[Henry (unit)|H]]  SI
cH  [[Henry (unit)|H]]  SI
mH  [[Henry (unit)|H]]  SI
µH  [[Henry (unit)|H]]  SI
uH  [[Henry (unit)|H]]  SI
nH  [[Henry (unit)|H]]  SI
pH  [[Henry (unit)|H]]  SI
fH  [[Henry (unit)|H]]  SI
```

```
aH  [[Henry (unit)|H]]  SI
zH  [[Henry (unit)|H]]  SI
yH  [[Henry (unit)|H]]  SI

A  [[Ampere|A]]  SI
YA  [[Ampere|A]]  SI
ZA  [[Ampere|A]]  SI
EA  [[Ampere|A]]  SI
PA  [[Ampere|A]]  SI
TA  [[Ampere|A]]  SI
GA  [[Ampere|A]]  SI
MA  [[Ampere|A]]  SI
kA  [[Ampere|A]]  SI
hA  [[Ampere|A]]  SI
daA  [[Ampere|A]]  SI
dA  [[Ampere|A]]  SI
cA  [[Ampere|A]]  SI
mA  [[Ampere|A]]  SI
µA  [[Ampere|A]]  SI
uA  [[Ampere|A]]  SI
nA  [[Ampere|A]]  SI
pA  [[Ampere|A]]  SI
fA  [[Ampere|A]]  SI
aA  [[Ampere|A]]  SI
zA  [[Ampere|A]]  SI
yA  [[Ampere|A]]  SI

V  [[Volt|V]]  SI
YV  [[Volt|V]]  SI
ZV  [[Volt|V]]  SI
EV  [[Volt|V]]  SI
PV  [[Volt|V]]  SI
TV  [[Volt|V]]  SI
GV  [[Volt|V]]  SI
MV  [[Volt|V]]  SI
kV  [[Volt|V]]  SI
hV  [[Volt|V]]  SI
daV  [[Volt|V]]  SI
dV  [[Volt|V]]  SI
cV  [[Volt|V]]  SI
mV  [[Volt|V]]  SI
µV  [[Volt|V]]  SI
uV  [[Volt|V]]  SI
nV  [[Volt|V]]  SI
pV  [[Volt|V]]  SI
fV  [[Volt|V]]  SI
aV  [[Volt|V]]  SI
zV  [[Volt|V]]  SI
yV  [[Volt|V]]  SI

VA  [[Volt-ampere|VA]]  SI
YVA  [[Volt-ampere|VA]]  SI
ZVA  [[Volt-ampere|VA]]  SI
EVA  [[Volt-ampere|VA]]  SI
PVA  [[Volt-ampere|VA]]  SI
TVA  [[Volt-ampere|VA]]  SI
GVA  [[Volt-ampere|VA]]  SI
MVA  [[Volt-ampere|VA]]  SI
kVA  [[Volt-ampere|VA]]  SI
hVA  [[Volt-ampere|VA]]  SI
daVA  [[Volt-ampere|VA]]  SI
dVA  [[Volt-ampere|VA]]  SI
cVA  [[Volt-ampere|VA]]  SI
mVA  [[Volt-ampere|VA]]  SI
```

```
µVA  [[Volt-ampere|VA]]  SI
uVA  [[Volt-ampere|VA]]  SI
nVA  [[Volt-ampere|VA]]  SI
pVA  [[Volt-ampere|VA]]  SI
fVA  [[Volt-ampere|VA]]  SI
aVA  [[Volt-ampere|VA]]  SI
zVA  [[Volt-ampere|VA]]  SI
yVA  [[Volt-ampere|VA]]  SI

Ω  [[Ohm|Ω]]  SI

YΩ.m  [[Electrical resistivity and conductivity#Definition|YΩ&sdot;m]]  1e24
ZΩ.m  [[Electrical resistivity and conductivity#Definition|ZΩ&sdot;m]]  1e21
EΩ.m  [[Electrical resistivity and conductivity#Definition|EΩ&sdot;m]]  1e18
PΩ.m  [[Electrical resistivity and conductivity#Definition|PΩ&sdot;m]]  1e15
TΩ.m  [[Electrical resistivity and conductivity#Definition|TΩ&sdot;m]]  1e12
GΩ.m  [[Electrical resistivity and conductivity#Definition|GΩ&sdot;m]]  1e9
MΩ.m  [[Electrical resistivity and conductivity#Definition|MΩ&sdot;m]]  1e6
kΩ.m  [[Electrical resistivity and conductivity#Definition|kΩ&sdot;m]]  1e3
Ω.m  [[Electrical resistivity and conductivity#Definition|Ω&sdot;m]]    1
mΩ.m  [[Electrical resistivity and conductivity#Definition|mΩ&sdot;m]]  1e-3
µΩ.m  [[Electrical resistivity and conductivity#Definition|µΩ&sdot;m]]  1e-6
uΩ.m  [[Electrical resistivity and conductivity#Definition|µΩ&sdot;m]]  1e-6
nΩ.m  [[Electrical resistivity and conductivity#Definition|nΩ&sdot;m]]  1e-9
pΩ.m  [[Electrical resistivity and conductivity#Definition|pΩ&sdot;m]]  1e-12
fΩ.m  [[Electrical resistivity and conductivity#Definition|fΩ&sdot;m]]  1e-15
aΩ.m  [[Electrical resistivity and conductivity#Definition|aΩ&sdot;m]]  1e-18
zΩ.m  [[Electrical resistivity and conductivity#Definition|zΩ&sdot;m]]  1e-21
yΩ.m  [[Electrical resistivity and conductivity#Definition|yΩ&sdot;m]]  1e-24

R  [[Rayleigh (unit)|R]]  SI

G  [[Gauss (unit)|G]]  SI
aG  [[Attogauss|G]]  SI
cG  [[Centigauss|G]]  SI
daG  [[Decagauss|G]]  SI
dG  [[Decigauss|G]]  SI
EG  [[Exagauss|G]]  SI
fG  [[Femtogauss|G]]  SI
GG  [[Gigagauss|G]]  SI
hG  [[Hectogauss|G]]  SI
kG  [[Kilogauss|G]]  SI
MG  [[Megagauss|G]]  SI
mG  [[Milligauss|G]]  SI
uG  [[Microgauss|G]]  SI
µG  [[Microgauss|G]]  SI
nG  [[Nanogauss|G]]  SI
PG  [[Petagauss|G]]  SI
pG  [[Picogauss|G]]  SI
TG  [[Teragauss|G]]  SI
yG  [[Yoctogauss|G]]  SI
YG  [[Yottagauss|G]]  SI
zG  [[Zeptogauss|G]]  SI
ZG  [[Zettagauss|G]]  SI

T  [[Tesla (unit)|T]]  SI
aT  [[Attotesla|T]]  SI
cT  [[Centitesla|T]]  SI
daT  [[Decatesla|T]]  SI
dT  [[Decitesla|T]]  SI
ET  [[Exatesla|T]]  SI
fT  [[Femtotesla|T]]  SI
GT  [[Gigatesla|T]]  SI
hT  [[Hectotesla|T]]  SI
```

```
kT  [[Kilotesla|T]]  SI
MT  [[Megatesla|T]]  SI
mT  [[Millitesla|T]]  SI
uT  [[Microtesla|T]]  SI
µT  [[Microtesla|T]]  SI
nT  [[Nanotesla|T]]  SI
PT  [[Petatesla|T]]  SI
pT  [[Picotesla|T]]  SI
TT  [[Teratesla|T]]  SI
yT  [[Yoctotesla|T]]  SI
YT  [[Yottatesla|T]]  SI
zT  [[Zeptotesla|T]]  SI
ZT  [[Zettatesla|T]]  SI

== Astrophysics ==
au  [[Astronomical unit|au]]
c  [[Speed of light|''c'']]
ly  [[Light-year|ly]]
dex  [[decimal exponent|dex]]
Earth mass  [[Earth mass|''M''<sub></sub>]]
Earth radius  [[Earth radius|''R''<sub></sub>]]
M_Earth  [[Earth mass|''M''<sub></sub>]]
R_Earth  [[Earth radius|''R''<sub></sub>]]
M+  [[Earth mass|''M''<sub></sub>]]
R+  [[Earth radius|''R''<sub></sub>]]
Jupiter mass  [[Jupiter mass|''M''<sub>J</sub>]]
Jupiter radius  [[Jupiter radius|''R''<sub>J</sub>]]
M_Jupiter  [[Jupiter mass|''M''<sub>J</sub>]]
R_Jupiter  [[Jupiter radius|''R''<sub>J</sub>]]
Solar mass  [[Solar mass|''M''<sub>&#x2609;</sub>]]
solar mass  [[Solar mass|''M''<sub>&#x2609;</sub>]]
M_Solar  [[Solar mass|''M''<sub>&#x2609;</sub>]]
M_solar  [[Solar mass|''M''<sub>&#x2609;</sub>]]
R_Solar  [[Solar radius|''R''<sub>&#x2609;</sub>]]
R_solar  [[Solar radius|''R''<sub>&#x2609;</sub>]]
Solar radius  [[Solar radius|''R''<sub>&#x2609;</sub>]]
solar radius  [[Solar radius|''R''<sub>&#x2609;</sub>]]
Solar luminosity  [[Solar luminosity|''L''<sub>&#x2609;</sub>]]
solar luminosity  [[Solar luminosity|''L''<sub>&#x2609;</sub>]]
L_solar  [[Solar luminosity|''L''<sub>&#x2609;</sub>]]
L_Solar  [[Solar luminosity|''L''<sub>&#x2609;</sub>]]
Lo  [[Solar luminosity|''L''<sub>&#x2609;</sub>]]
pc2  [[Parsec|pc<sup>2</sup>]]
pc3  [[Parsec|pc<sup>3</sup>]]
kpc2  [[Parsec#Parsecs and kiloparsecs|kpc<sup>2</sup>]]
kpc3  [[Parsec#Parsecs and kiloparsecs|kpc<sup>3</sup>]]
kpc  [[Parsec#Parsecs and kiloparsecs|kpc]]
Mpc2  [[Parsec#Megaparsecs and gigaparsecs|Mpc<sup>2</sup>]]
Mpc3  [[Parsec#Megaparsecs and gigaparsecs|Mpc<sup>3</sup>]]
Mpc  [[Parsec#Megaparsecs and gigaparsecs|Mpc]]
Gpc2  [[Parsec#Megaparsecs and gigaparsecs|Gpc<sup>2</sup>]]
Gpc3  [[Parsec#Megaparsecs and gigaparsecs|Gpc<sup>3</sup>]]
Gpc  [[Parsec#Megaparsecs and gigaparsecs|Gpc]]

== Nuclear physics and chemistry ==
cm-1  [[Wavenumber|cm<sup>−1</sup>]]
u  [[Unified atomic mass unit|u]]
osmol  [[Osmole (unit)|osmol]]
Osm  [[Osmole (unit)|Osm]]
M  [[Molarity|M]]
TM  [[Molarity|M]]  SI
GM  [[Molarity|M]]  SI
MM  [[Molarity|M]]  SI
kM  [[Molarity|M]]  SI
```

```
hM  [[Molarity|M]]  SI
daM  [[Molarity|M]]  SI
dM  [[Molarity|M]]  SI
cM  [[Molarity|M]]  SI
mM  [[Molarity|M]]  SI
uM  [[Molarity|M]]  1e-6
nM  [[Molarity|M]]  SI
pM  [[Molarity|M]]  SI
kg.mol-1  [[Molar mass|kg&sdot;mol<sup>−1</sup>]]
kg/mol  [[Molar mass|kg/mol]]
g.mol-1  [[Molar mass|g&sdot;mol<sup>−1</sup>]]
g/mol  [[Molar mass|g/mol]]
eV/c2  [[Electronvolt#Mass|eV/''c''<sup>2</sup>]]
keV/c2  [[Electronvolt#Mass|keV/''c''<sup>2</sup>]]
MeV/c2  [[Electronvolt#Mass|MeV/''c''<sup>2</sup>]]
GeV/c2  [[Electronvolt#Mass|GeV/''c''<sup>2</sup>]]
TeV/c2  [[Electronvolt#Mass|TeV/''c''<sup>2</sup>]]
μN  [[Nuclear magneton|μ<span style="display:inline-block;margin-bottom:-0.3em;ve
μB  [[Bohr magneton|μ<span style="display:inline-block;margin-bottom:-0.3em;verti
eV  [[Electronvolt|eV]]
meV  [[Electronvolt|meV]]
keV  [[Electronvolt|keV]]
MeV  [[Electronvolt|MeV]]
GeV  [[Electronvolt|GeV]]
TeV  [[Electronvolt|TeV]]
mol-1  [[Avogadro constant|mol<sup>−1</sup>]]
J.mol-1  [[Joule per mole|J&sdot;mol<sup>−1</sup>]]
J/mol  [[Joule per mole|J/mol]]
kJ.mol-1  [[Joule per mole|kJ&sdot;mol<sup>−1</sup>]]
kJ/mol  [[Joule per mole|kJ/mol]]
MJ.mol-1  [[Joule per mole|MJ&sdot;mol<sup>−1</sup>]]
MJ/mol  [[Joule per mole|MJ/mol]]
GJ.mol-1  [[Joule per mole|GJ&sdot;mol<sup>−1</sup>]]
GJ/mol  [[Joule per mole|GJ/mol]]
TJ.mol-1  [[Joule per mole|TJ&sdot;mol<sup>−1</sup>]]
TJ/mol  [[Joule per mole|TJ/mol]]

== Numbers and phrases ==
pp  [[Page (paper)|pp]]
ppb  [[Parts per billion|ppb]]  1e-9
ppm  [[Parts per million|ppm]]  1e-6
billiard  [[Orders of magnitude (numbers)#1015|billiard]]  1e15
billion  [[1,000,000,000|billion]]  1e9
billionth  [[1,000,000,000|billionth]]  1e-9
billionths  [[1,000,000,000|billionths]]  1e-9
decilliard  [[Orders of magnitude (numbers)#1063|decilliard]]  1e63
decillion  [[Orders of magnitude (numbers)#1033|decillion]]  1e33
decillionth  [[Orders of magnitude (numbers)#1033|decillionth]]  1e-33
decillionths  [[Orders of magnitude (numbers)#1033|decillionths]]  1e-33
milliard  [[1,000,000,000|milliard]]  1e9
million  [[Million|million]]  1e6
millionth  [[Million|millionth]]  1e-6
millionths  [[Million|millionths]]  1e-6
nonilliard  [[Orders of magnitude (numbers)#1057|nonilliard]]  1e57
nonillion  [[Orders of magnitude (numbers)#1030|nonillion]]  1e30
nonillionth  [[Orders of magnitude (numbers)#1030|nonillionth]]  1e-30
nonillionths  [[Orders of magnitude (numbers)#1030|nonillionths]]  1e-30
octilliard  [[Orders of magnitude (numbers)#1051|octilliard]]  1e51
octillion  [[Orders of magnitude (numbers)#1027|octillion]]  1e27
octillionth  [[Orders of magnitude (numbers)#1027|octillionth]]  1e-27
octillionths  [[Orders of magnitude (numbers)#1027|octillionths]]  1e-27
quadrilliard  [[Orders of magnitude (numbers)#1027|quadrilliard]]  1e27
quadrillion  [[Orders of magnitude (numbers)#1015|quadrillion]]  1e15
quadrillionth  [[Orders of magnitude (numbers)#1015|quadrillionth]]  1e-15
```

```
quadrillionths  [[Orders of magnitude (numbers)#1015|quadrillionths]]  1e-15
quintilliard  [[Orders of magnitude (numbers)#1033|quintilliard]]  1e33
quintillion  [[Orders of magnitude (numbers)#1018|quintillion]]  1e18
quintillionth  [[Orders of magnitude (numbers)#1018|quintillionth]]  1e-18
quintillionths  [[Orders of magnitude (numbers)#1018|quintillionths]]  1e-18
septilliard  [[Orders of magnitude (numbers)#1045|septilliard]]  1e45
septillion  [[Orders of magnitude (numbers)#1024|septillion]]  1e24
septillionth  [[Orders of magnitude (numbers)#1024|septillionth]]  1e-24
septillionths  [[Orders of magnitude (numbers)#1024|septillionths]]  1e-24
sextilliard  [[Orders of magnitude (numbers)#1039|sextilliard]]  1e39
sextillion  [[Orders of magnitude (numbers)#1021|sextillion]]  1e21
sextillionth  [[Orders of magnitude (numbers)#1021|sextillionth]]  1e-21
sextillionths  [[Orders of magnitude (numbers)#1021|sextillionths]]  1e-21
trilliard  [[Orders of magnitude (numbers)#1021|trilliard]]  1e21
trillion  [[Orders of magnitude (numbers)#1012|trillion]]  1e12
trillionth  [[Orders of magnitude (numbers)#1012|trillionth]]  1e-12
trillionths  [[Orders of magnitude (numbers)#1012|trillionths]]  1e-12

== Angles ==
%                   %                                       Percent
percent             %                                       Percent
per cent            %                                       Percent
‰                   ‰                                       Per mil
per mil             ‰                                       Per mil
per mill            ‰                                       Per mil
per mille           ‰                                       Per mil
permil              ‰                                       Per mil
permill             ‰                                       Per mil
permille            ‰                                       Per mil
°                   °                                       Degree (angle)
deg                 °                                       Degree (angle)
'                   ′                                       Minute of arc
′                   ′                                       Minute of arc
arcmin              ′                                       Minute of arc
arcminute           ′                                       Minute of arc
"                   ″                                       Second of arc
″                   ″                                       Second of arc
arcsec              ″                                       Second of arc
arcsecond           ″                                       Second of arc
mas  [[Milliarcsecond|mas]]  pi/648000000

]=]

-- If val has "|long scale=on" the following definitions are used
-- (then, if not found here, the normal definitions are used).
-- Unit code  [[Link|Symbol]]  Flags/Scale
local builtin_units_long_scale = [=[

== Long scale numbers and phrases ==
billion  [[Orders of magnitude (numbers)#1012|billion]]  1e12
billionth  [[Orders of magnitude (numbers)#1012|billionth]]  1e-12
billionths  [[Orders of magnitude (numbers)#1012|billionths]]  1e-12
decillion  [[Orders of magnitude (numbers)#1060|decillion]]  1e60
decillionth  [[Orders of magnitude (numbers)#1060|decillionth]]  1e-60
decillionths  [[Orders of magnitude (numbers)#1060|decillionths]]  1e-60
nonillion  [[Orders of magnitude (numbers)#1054|nonillion]]  1e54
nonillionth  [[Orders of magnitude (numbers)#1054|nonillionth]]  1e-54
nonillionths  [[Orders of magnitude (numbers)#1054|nonillionths]]  1e-54
octillion  [[Orders of magnitude (numbers)#1048|octillion]]  1e48
octillionth  [[Orders of magnitude (numbers)#1048|octillionth]]  1e-48
octillionths  [[Orders of magnitude (numbers)#1048|octillionths]]  1e-48
quadrillion  [[Orders of magnitude (numbers)#1024|quadrillion]]  1e24
quadrillionth  [[Orders of magnitude (numbers)#1024|quadrillionth]]  1e-24
quadrillionths  [[Orders of magnitude (numbers)#1024|quadrillionths]]  1e-24
```

```
quintillion  [[Orders of magnitude (numbers)#1030|quintillion]]  1e30
quintillionth  [[Orders of magnitude (numbers)#1030|quintillionth]]  1e-30
quintillionths  [[Orders of magnitude (numbers)#1030|quintillionths]]  1e-30
septillion  [[Orders of magnitude (numbers)#1042|septillion]]  1e42
septillionth  [[Orders of magnitude (numbers)#1042|septillionth]]  1e-42
septillionths  [[Orders of magnitude (numbers)#1042|septillionths]]  1e-42
sextillion  [[Orders of magnitude (numbers)#1036|sextillion]]  1e36
sextillionth  [[Orders of magnitude (numbers)#1036|sextillionth]]  1e-36
sextillionths  [[Orders of magnitude (numbers)#1036|sextillionths]]  1e-36
trillion  [[Orders of magnitude (numbers)#1018|trillion]]  1e18
trillionth  [[Orders of magnitude (numbers)#1018|trillionth]]  1e-18
trillionths  [[Orders of magnitude (numbers)#1018|trillionths]]  1e-18

]=]

return { builtin_units = builtin_units, builtin_units_long_scale = builtin_units_
```