Timeframes

Introduction

The *timeframe* of a chart is sometimes also referred to as its *interval* or *resolution*. It is the unit of time represented by one bar on the chart. All standard chart types use a timeframe: “Bars”, “Candles”, “Hollow Candles”, “Line”, “Area” and “Baseline”. One non-standard chart type also uses timeframes: “Heikin Ashi”.

Programmers interested in accessing data from multiple timeframes will need to become familiar with how timeframes are expressed in Pine Script™, and how to use them.

Timeframe strings come into play in different contexts:

- They must be used in request.security() when requesting data from another symbol and/or timeframe. See the page on Other timeframes and data to explore the use of request.security().
- They can be used as an argument to time() and time_close() functions, to return the time of a higher timeframe bar. This, in turn, can be used to detect changes in higher timeframes from the chart’s timeframe without using request.security(). See the Testing for changes in higher timeframes section to see how to do this.
- The input.timeframe() function provides a way to allow script users to define a timeframe through a script’s “Inputs” tab (see the Timeframe input section for more information).
- The indicator() declaration statement has an optional `timeframe` parameter that can be used to provide multi-timeframe capabilities to simple scripts without using request.security().
- Many built-in variables provide information on the timeframe used by the chart the script is running on. See the Chart timeframe section for more information on them, including timeframe.period which returns a string in Pine Script™’s timeframe specification format.

Timeframe string specifications

Timeframe strings follow these rules:

- They are composed of the multiplier and the timeframe unit, e.g., “1S”, “30” (30 minutes), “1D” (one day), “3M” (three months).
- The unit is represented by a single letter, with no letter used for minutes: “S” for seconds, “D” for days, “W” for weeks and “M” for months.
- When no multiplier is used, 1 is assumed: “S” is equivalent to “1S”, “D” to “1D, etc. If only “1” is used, it is interpreted as “1min”, since no unit letter identifier is used for minutes.

- There is no “hour” unit; “1H” is **not** valid. The correct format for one hour is “60” (remember no unit letter is specified for minutes).
- The valid multipliers vary for each timeframe unit:
    - For seconds, only the discrete 1, 5, 10, 15 and 30 multipliers are valid.
    - For minutes, 1 to 1440.
    - For days, 1 to 365.
    - For weeks, 1 to 52.
    - For months, 1 to 12.

Comparing timeframes

It can be useful to compare different timeframe strings to determine, for example, if the timeframe used on the chart is lower than the higher timeframes used in the script, as using timeframes lower than the chart is usually not a good idea. See the Requesting data of a lower timeframe section for more information on the subject.

Converting timeframe strings to a representation in fractional minutes provides a way to compare them using a universal unit. This script uses the timeframe.in_seconds() function to convert a timeframe into float seconds and then converts the result into minutes:

```
//@version=5
indicator("Timeframe in minutes example", "", true)
string tfInput = input.timeframe(defval = "", title = "Input TF")

float chartTFInMinutes = timeframe.in_seconds() / 60
float inputTFInMinutes = timeframe.in_seconds(tfInput) / 60

var table t = table.new(position.top_right, 1, 1)
string txt = "Chart TF: "    + str.tostring(chartTFInMinutes, "#.##### minutes") +
"\nInput TF: " + str.tostring(inputTFInMinutes, "#.##### minutes")
if barstate.isfirst
    table.cell(t, 0, 0, txt, bgcolor = color.yellow)
else if barstate.islast
    table.cell_set_text(t, 0, 0, txt)

if chartTFInMinutes > inputTFInMinutes
    runtime.error("The chart's timeframe must not be higher than the input's timeframe.
```

Note that:

- We use the built-in timeframe.in_seconds() function to convert the chart and the input.timeframe() function into seconds, then divide by 60 to convert into minutes.
- We use two calls to the timeframe.in_seconds() function in the initialization of the `chartTFInMinutes` and `inputTFInMinutes` variables. In the first instance, we do not supply an argument for its `timeframe` parameter, so the function returns the chart’s timeframe in seconds. In the second call, we supply the timeframe selected by the script’s user through the call to input.timeframe().
- Next, we validate the timeframes to ensure that the input timeframe is equal to or higher than the chart’s timeframe. If it is not, we generate a runtime error.
- We finally print the two timeframe values converted to minutes.

Options v: v5