Quick and Dirty Daylight Indicator

The following is a function to provide an indicator of whether the given time occurs during “Civil Daylight” on the given date. This is the light period of the day considered between the twilights of sunrise and sunset. To do this with precision requires horrible, complex (= slow) maths and it's a huge overkill for a simple Go/NoGo test.

This function was written to provide some intelligence to exterior lighting so that it is only activated/enabled during the night, thus precision is not required - within 5 or 10 minutes either way is fine.

On the downside: It is very rough and ready: The sunrise/set times change considerably between the start and finish of say, March. This function uses the start of the current month and the start of the next to create a linear progression based around 30 divisions (days). If you graphed it, it wouldn't be a smooth sine wave over the year but rather an analogue made up of a series of straight lines (with some jaggies around month change because for simplicity it assumes there are 30 days in every month). The use of Civil Daylight means there is a tendency to over-compensate for darkness anyway so it shouldn't be a problem.

On the upside: Because it is very maths light, it is really quick, <4ms @ 48MHz so it doesn't cost much if you want to call it from the main loop.

The numbers in the first `Case` statement are the STart and FiNish of daylight as minutes from midnight for the first day of the relevant month with no daylight saving time. I did try this with the times on the 22nd of each month so the function would be exactly right on the longest and shortest days of the year but it resulted in a much higher fluctuation over the remainder of the month due to the 22 day offset early on in each month and only 8 days after.

A day contains 1440 minutes and so daylight may start at the 405th minute and continue until the 1060th minute (i.e. 06:45 until 17:40). This sunrise/set online tool was used to obtain the daylight times with a few tweaks. Converting to the numbers here is simply the first day of each month, and I based them on civil twilight start and stop times; HH*60+MM. There are 13 groups; having an additional month for January after December removes the need for a modulo and so is a tiny speed increase at the expense of a bit of program space. The times shown are for London, UK rounded up or down to 5 minutes - you can change these for your own location but do not include daylight saving times.

This version includes an optional parameter to return the start and finish times of daylight also: Opt=1 returns the start of daylight in minutes since midnight on the given date. Opt=2 returns the end of daylight.

Requires a single string which is a concatenation of the standard DATE$ and TIME$ strings as argument - see the Now() Function for an easy interface. NOTE does not support the international date sort form (yyyy-mm-dd…) - yet.

Syntax:

```
x=IsDayLight(strDateTime,opt)
```

Do not use Daylight saving datetime format is “dd-mm-yyyy hh:mm:ss”

Example Uses:

```
DaylightAug5am=IsDayLight("17/08/2017 05:00:00")
```

```
If IsDayLight(Now()) Then LightsOff Else LightsOn
```

```
Print IsDayLight("12/10/2020 15:30:00",1)
```

show the start time of daylight (in minutes since midnight) for the given date

```
Print IsDayLight(Now(),2)
```

show the end time of daylight

Dependancieis: DatePart()

Code

```
    Function IsDayLight(dt$,opt as integer) As Integer
        'opt = 1, return daytime start, 2, return daytime end anything else
return bool of daytime now
        Local Integer mn,st,fn,st0,fn0,dd,mm
        dd=DatePart("dd",dt$):mm=DatePart("mm",dt$)
        Do
            Select Case mm
                Case 1:st=490:fn=960
                Case 2:st=460:fn=1010
                Case 3:st=405:fn=1060
                Case 4:st=335:fn=1115
                Case 5:st=275:fn=1165
                Case 6:st=230:fn=1210
                Case 7:st=230:fn=1220
                Case 8:st=265:fn=1190
                Case 9:st=315:fn=1115
                Case 10:st=360:fn=1060
                Case 11:st=415:fn=990
                Case 12:st=470:fn=955
                Case 13:st=490:fn=960
            End Select
            If fn0<>0 Then Exit Do
            fn0=fn:st0=st
            mm=mm+1
        Loop
        st=st0-(((st0-st)/30)*dd):fn=fn0-(((fn0-fn)/30)*dd)
        Select Case opt
            Case 1
                IsDayLight=st
            Case 2
                IsDayLight=fn
```

```
            Case Else
                mn=(DatePart("h",dt$)*60)+DatePart("m",dt$)
                IsDayLight=(mn>=st) And (mn<fn)
        End Select
    End Function
```

Mins2Time

Function to convert minutes from midnight (0 - 1439) to an actual HH:MM:SS time. Primarily intended for converting values returned by options 1 & 2 above. No error checking.

Dependencies: ZPad$()

Syntax

=Mins2Time(n)

Examples

Print “Civil daylight today starts at ”;Mins2Time(IsDayLight(Now(),1))
SunSet$=Mins2Time(IsDayLight(Now(),2))

```
    Function Mins2Time(t As Integer) As String
        Local Integer h
        h=(t\60)
        Mins2Time=ZPad$(h,2)+":"+ZPad$(t-(h*60),2)+":00"
    End Function
```