IsDST - Function to Test a DateTime to see if Daylight Savings should be applied

System clocks are best run at a local time that isn't influenced by Daylight Savings Time (DST). In the UK, we are fortunate enough to have our basic time as UTC.

Having times move about can be quite disconcerting, especially when logging. Seeing several entries with the same time around the end of October can cause quite a bit of confusion. Consequently, it is quite usual to run systems on a time base that does not include DST - this could be GMT or your own flavour, or even UTC, however UTC can cause difficulty when the local time is considerably different.

When using a local time without DST, actually displaying local time can also be confusing in the DST period when the clock will appear to be an hour slow and setting alarms etc is problematic because the hour difference is often forgotten.

Below is a Function (and its attendant support functions) that will calculate a simple boolean (actually an integer) answer for if a given DateTime should be subject to DST. So now you can run all your system times in GMT or whatever and only add-on the DST when you want to display it to fallible humans.

Daylight saving time here in the UK is termed BST (British Summer Time) and is one hour ahead of GMT from 02:00 on the last Sunday of March until 02:00 on the last Sunday of October. The problem is that these Sundays always move about. The Function takes this into account with several shortcuts, only doing calculation when it is necessary.

The support Functions include a modified form of the Day-of-Week function found elsewhere in this section.

For accurate function, it is necessary to pass DateTimes that do not already have a DST component.

A brief explanation of the algorithm: Months 4,5,6,7,8,9 are subject to DST, so if the given date is in one of them we quickly return a 1 (DST applies). Likewise, all dates in months 1,2,11,12 do not have DST so we return a 0 (DST does not apply). In UK, DST changes on the last Sunday of months 3 and 10 and it follows that must occur sometime in the last week of those months; each of which has 31 days. 31-7 = 24, thus If the date is before the 25th we can quickly deduce that DST does or does not apply. For the remaining week in those months, we seek the Sunday by taking the 25th day (the first that could be a Sunday) and determining its day of the week number. We subtract this from 7 and the result is the number of days we must add to jump to the real date of the Sunday. It is then a simple case comparing the unixtimes of 2am on that Sunday with the supplied date (using unixtimes avoids all the faff associated with date strings).

If the points at which DST for your region are different, it shouldn't be too hard to tweak the function to your own requirements.

Dependencies:

UnixTime or Epoch Time

Syntax:

=IsDST(DateTime)

Examples:

If IsDST(“01-01-1970 00:00:00”) Then … HumanDate$=DateAdd(IsDST(Now(),“h”,Now()))'convert UTC to local UK time

```
    Function IsDST(d$) As Integer
        Select Case Val(Mid$(d$,4,2))' extract the month number
            Case 4 To 9' summer months, early bath
                IsDST=1
            Case 1,2,11,12' winter months, early bath
                IsDST=0
            Case 3
                If Val(Left$(d$,2))<25 Then' still winter, early bath
                    IsDST=0
                Else
                    IsDST=Not(UnixTime(d$)<UnixTime(FindLastSunday(d$)))
'compare the two unixtimes
                EndIf
            Case 10
                If Val(Left$(d$,2))<25 Then' still summer, early bath
                    IsDST=1
                Else
                    IsDST=(UnixTime(d$)<UnixTime(FindLastSunday(d$)))
'compare the two unixtimes
                EndIf
        End Select
    End Function
    Function FindLastSunday(d$) As String
        Local a$
        a$="25"+Mid$(d$,3,9)+"02:00:00"' starting from midnight on the 25th
- the earliest possible Sunday (with embedded time when DST changes)
        FindLastSunday=Str$(25+((7-Dow(a$)) Mod 7),2,0,"0")+Mid$(a$,3)' add
the difference in the day number and generate the true change datetime
    End Function
    'Day of week for given date, 0=Sun
    Function Dow(dt$) As Integer
        Local Integer m,d,y
        d=Val(Left$(dt$,2))
        m=Val(Mid$(dt$,4,2))
        y=Val(Mid$(dt$,7,4))
        If m<3 Then
            m=m+12
            y=y-1
        End If
        dow=(d+((26*m+1)/10)-(2*(y\100))+y+(y\4)+(y\400)) Mod 7
    End Function
```

See Also:

Now() Function (VB work_a_like)