**Date and Time**

**Chapter Twelve**

Excel already has an extensive range of functions which manipulate date and time. As well as being able to customize this functionality, VBA can extend it to provide extra capability.

**Covers**

Now, Date and Time 172

Separating date components 173

Assembling date components 174

Adding and subtracting dates 175

The number of weekdays 176

The Mod operator 177

Anniversaries 178

 YouTube videos:

 VBACh12a.avi [https://youtu.be/0gJJ7if5dk0](https://youtu.be/0gJJ7if5dk0%22%20%5Ct%20%22_blank) anomaly Now() Format() Date Time

 VBACh12b.avi [https://youtu.be/5ukjo8LhTYc](https://youtu.be/5ukjo8LhTYc%22%20%5Ct%20%22_blank) DateValue/## Part/Serial/Diff

 Year Month Day Hour Minute Second DatePart DateSerial

 Exercise 1st day of this month DateAdd DateDiff

 VBACh12c.avi [https://youtu.be/UadqvMrVJmo](https://youtu.be/UadqvMrVJmo%22%20%5Ct%20%22_blank) Days between. Weekdays BDay How Old Day of birth Mod Leap Year

**Date and Time**

Microsoft date specifies the number of days elapsed since 1900 plus the fraction of a day eg.



If we format the number as a custom date which has time as well:



As was intended day 1 is 1/1/1900. Bit this implied that day 0 is the day before ie 31/12/199.

So so far the Microsoft date system starts at 30/12/1899.

But we haven’t finished!



Place 0 in cell A2. Ctrl-drag down to about 100. Copy this column across to column B and format column B as Date.) You may need to repair cell B2 and change it from 00/01/1900 to 31/12/1899.

But someone made a mistake! Take a look further down to day 60.



Somebody thought that 1900 was a leap year. wrong! This day did not exist.

Microsoft blames Lotus 123 for this – or put differently “there were so many s/sheets in existence with the wrong date” that they didn’t want to change it. mmm…

If we wish it to be the 61st day move the goalposts! ie make the start date one day earlier: 30th Dec 1899.

To illustrate this you could select all the dates up to and including 59 28/02/1900 and drag them all down by one (OVER the 60 29/02/1900 ) and then make day 0 into 29/12/1899.

The dates before 28/02/1900 will then not be correct!

So we finally have it. The official M/soft start date is 30th Dec 1899 – day 0 (00:am).

Whereas this may seem at first sight a rather drastic error, (days before day 61 will be wrong it is not as bad as it may seem since are usually concerned with date *differences* after that bogus day.

# Now, Date and Time

## Now

*Even though Now() is a function, you may or may not include*

*parentheses, i.e. MsgBox Now*

Now specifies the current date and time.

Private Sub CommandButton1\_Click() MsgBox Now()

End Sub

*will suffice.* The format of the date actually displayed, US or otherwise, will depend upon the Windows Regional settings.

To explicitly format a Date, use the Format function, e.g. MsgBox Format (Now, "mmmm") would display “September” in this case and MsgBox Format (Now, "mmm") would display “Sep”. (See VBA Help on the Format function.)

*The original intention was to make Jan 1 1900*

*day 1, i.e. Dec 31*

*1899 would be day 0, but unfortunately, somebody included Feb 29 in1900 (not a leap year – see later), so Dec 30 1899 was subsequently made the start date.*

Excel stores dates internally as a Date type (which is of the same 8-byte form as a Double type), the integral part of which is the number of days since Dec 30 1899, and the decimal part of which represents the fraction of the day after midnight.

Although dates are stored as decimal numbers, the way that we view them depends upon how they are formatted. We can force a type conversion to a Double data type using:

Private Sub CommandButton1\_Click() MsgBox CDbl(Now)

End Sub

Number of days since Dec 30, 1899.

Fraction of the day since midnight.

## Date and Time

The Date function returns only the current date without the time,

e.g. MsgBox Date would display “09/09/2004” or similar, (again the format displayed will depend upon the Windows Regional setting) whereas MsgBox Time would return “15:53:20” or similar.

*Without any delimiters, Excel VBA will perform a division, e.g.*

*MsgBox 1/1/ 2000 will cause .0005 to be displayed.*

*CDate is even more versatile than DateValue. It will accept numerical*

*arguments as well as Strings.*

<https://youtu.be/5ukjo8LhTYc>

# Initializing a Date

# Date Delimiters & DateValue

The number sign ( # ) can be used as a delimiter to enclose a Date type, e.g. # 10/30/98 # , but it must conform to American format (mm/dd/yy). If it obviously doesn’t, it will be changed before your eyes! For example, if MsgBox Day(#30/10/98#) is typed, it will immediately change. Due to this ambiguity, date literals are best avoided. Using double quotes as a delimiter, e.g. “10/30/98” also

introduces ambiguity, as well as requiring an extra step to convert it to a date. To convert a String to a Date, use DateValue, e.g.

DateValue("December 1, 2004"). As well as being preferable to using date literals, DateValue is forgiving of the format of the String, e.g. MsgBox DateValue("Dec 1, 2004") will display the date in a format determined by the Regional settings.

# Separating date components

We can use the correspondingly-named functions to extract the required parts as Integer values. Examples are given below.

## Year, Month, Day

Private Sub CommandButton1\_Click() MsgBox Year(Date)

End Sub

## Hour, Minute, Second

Private Sub CommandButton1\_Click() MsgBox Second(Time)

End Sub

Of course we could have used Now as an argument in both cases instead of Date and Time respectively.

## DatePart

DatePart performs identically to the above functions, e.g.

DatePart("yyyy",Date) would also extract the year as did

*DatePart is also useful for working with year quarters, e.g.*

*DatePart(“q”, Date) will return the present numeric quarter (1, 2, 3 or 4).*

Year(Date) above. It does have some other uses however, for example to find the day of the year:

Private Sub CommandButton1\_Click() MsgBox DatePart("y",Date)

End Sub

Today is the 253rd

day of this year.

## DateValue, TimeValue

These extract just the date and time components respectively, e.g.

Private Sub CommandButton1\_Click() MsgBox DateValue(Now)

End Sub

The date part only of the current date/time is extracted. (Identical to MsgBox Date.)

Similarly, TimeValue would extract only the time part.

# Assembling date components

## DateSerial

Whereas Year, Month, etc. split a date into its components, DateSerial does the opposite. For example, DateSerial(2004,12,1) would form the date Dec 1, 2004.

If a message box were used to output such a date, it would be presented according to your particular Windows Regional settings, i.e. either 1/12/2004 or 12/1/2004.

Similarly, MsgBox TimeSerial (1, 2, 3) would display:

3 seconds after 2

minutes past 1.

# Combining them

 We wish to combine some of the previous operations

eg to find the date of first day of the current month use:

MsgBox DateSerial(Year(Date), Month(Date),1)

 Day 1

(The present date is 9 Sept 2004.)

The date of the first day of the month of Sept 2004 is. (Displayed here in US format.).

# Adding and subtracting dates

Since the integer portion of a Date type represents the number of days, we can simply add or subtract an integral number of days to find another date, e.g. to find tomorrow’s date we could use:

MsgBox DateValue(Now + 1)

## DateAdd

To add intervals other than days, we must use DateAdd. For example, to work with weeks, use the “ww” interval code and to display the date one week hence use:

MsgBox DateAdd("ww", 1, Date)

*The DateAdd interval code is “m” for months and “n” for minutes.*

*In the case of days, it would be possible to simply subtract the DateValues to*

*find the difference.*

(See the VBE Help for the other interval codes for DateAdd.)

## DateDiff

We can use DateDiff to find the number of days between two Dates. As well as taking into account the different lengths of the various months, DateDiff also takes into consideration the presence or absence of leap days between the two date intervals.

**Exercise: Find the number of days between Dec 1,2004 and Mar 1,2005**

Private Sub CommandButton1\_Click() Dim strtDate As Date, fnshDate As Date Dim n As Integer

strtDate = DateValue("Dec 1,2004") fnshDate = DateValue("Mar 1,2005")

n = DateDiff("d", strtDate, fnshDate) MsgBox n

End Sub

The number of days between 00:00 hrs Dec 1, 2004 and

00:00 hrs "Mar 1, 2005" is 90.

DateDiff can be also be used to find the number of months between two dates for example by changing the interval specifier to "m", e.g. DateDiff("m", strtDate, fnshDate).(See Help on the DateDiff function for the other interval specifiers.)

 <https://youtu.be/UadqvMrVJmo>

#  The number of weekdays

 **Exercise: Find the number of weekdays (working days) between Dec 1,2004 and Mar 1,2005**

## WeekDay

WeekDay will determine the day of the week as a number between 1 and 7 – where Sunday is 1 by default, e.g.

Private Sub CommandButton1\_Click() MsgBox Weekday(Date)

End Sub

In this case, today is Thursday - the 5th day of the week (with Sunday as day 1).

*We could omit DateDiff and modify the code, including the loop For i = strtDate*

We previously found the number of days between Dec 1 2004 and March 1 2005 to be 90. We now wish to find the number of *working* days (Monday to Friday) – perhaps for a project – between these two dates. (March 1 2005 itself will this time be included.)

Private Sub CommandButton1\_Click() Dim strtDate As Date, fnshDate As Date Dim dt As Date, n As Long, i As Long Dim numWorkDays As Long

strtDate = DateValue("Dec 1,2004") fnshDate = DateValue("Mar 1,2005") numWorkDays = 0

n = DateDiff("d", strtDate, fnshDate)

For i = 0 To n 'n since March 1 is included

dt = Weekday(strtDate + i)

*to fnshDate, since these date values are integral.*

If (dt <> 7 And dt <> 1) Then numWorkDays = numWorkDays + 1

If not a Sat and

not a Sun then...

*Use VB built-in constants wherever possible to make your*

End If Next i

MsgBox numWorkDays

End Sub

...increment the number of workdays.

*code more readable e.g. vbSunday and vbSaturday instead of 1 and 7.*

*The Internet is an invaluable source of VBA examples. Try typing “Excel VBA Date and*

*Time” into your search engine.*

The number of working days between project start and finish (inclusive) is 65.

As it stands, our program does not take account of any public

holidays in the period, e.g. Christmas day. These would need to be explicitly, individually accounted for by the code.

# The Mod operator

Mod gives the remainder after a division.

*Try MsgBox (1900 Mod 4 =*

*0) to confirm that it displays True.*

*We could also determine if a year is leap by testing if Month(DateSerial(yr,*

*2, 29)) = 2, (which seems to be forgiving of the bogus 1900 leap year!)*

*And will evaluate to False if either*

In this example 13 is divided by 3 (4 times) to give a remainder of 1 which would be displayed.

Private Sub CommandButton1\_Click() MsgBox 13 Mod 3

End Sub

Mod is an arithmetic operator, not a Date function but it could be used for example, to determine whether a year is leap or not. Mod can be used to determine whether a number is exactly divisible, e.g. 1900 Mod 4 would return 0, indicating that 1900 was exactly

divisible by 4.

This would suggest that the year 1900 was a leap year, since it is divisible by 4, but in addition, in 1582, Pope Gregory decreed, that years divisible by 100 were *not* to be leap years. If a variable yr were 1900 or 2000 for example, also stipulating that (yr Mod 100)

<> 0, i.e. yr is not divisible by 100, would rule out 1900 and 2000 since they are.

However, to further compensate for the earth’s non-exact rotation, the good Pope decreed that every 400 years *was* to be a leap year,

i.e. if we also allow the condition yr Mod 400 = 0 (i.e. yr could be divisible by 400), 2000 would now qualify, but 1900 would not. Combining these conditions into one logical expression, we could test for a leap year (e.g. 1900) using:

Private Sub CommandButton1\_Click()

*expression evaluates to*

Dim yr As Integer

yr = 1900

True for 1900.

*False. Or will evaluate to True if either expression evaluates to True. (Try MsgBox (True And False Or*

MsgBox (yr Mod 4 = 0) And (yr Mod 100 <> 0) \_

Or (yr Mod 400 = 0)

End Sub False for 1900

*False) to confirm that it displays False.)*

False for 1900, since 1900 is

not divisible by 400, i.e. 1900

since 1900 is

divisible by 100.

*To test the code further, try changing yr to 1901, 1904,*

*2000, etc.*

does not qualify on this basis. 1900 is not a leap year.

# Anniversaries

**Exercise**: **Given you birthdate and the date now determine how old you are.**

You might think that to find your age, you could simply subtract the year of your birth from the present year.

The only problem is that you may or may not have had your birthday yet this year. If you haven’t had your birthday, you will need to subtract a year. How do we determine if you have had your birthday this year or not? We need to test whether today’s date (Date) is less than the date of your birthday this year.

The date of your birthday this year will be given by DateSerial(Year(Date), Month(dob), Day(dob)), where dob is your date of birth, (i.e. your date of birth with the current year substituted for your year of birth) so

Date < DateSerial(Year(Date), Month(dob), Day(dob))

will be True if your birthday has yet to come this year.

This program should display your age if you use your date of birth.

Private Sub CommandButton1\_Click() Dim age As Integer

Dim dob As Date, bday As Date dob = DateValue("Oct 30, 1966")

bday = DateSerial(Year(Date), Month(dob), Day(dob)) age = Year(Date) - Year(dob) + CInt(Date < bday)

MsgBox age & " years old"

*True has the Integer value of*

*-1, whereas False has the Integer value of 0. (Try*

*MsgBox CInt(False) and MsgBox CInt(True) to confirm this.)*

End Sub

(Date < bday) will have the numeric value of -1 (True) if your birthday has not yet passed, and 0 (False) if it has.

Speaking of birthdays, you could determine the day of the week that you were born using this code with your date of birth.

Private Sub CommandButton1\_Click()

Dim dob As Date

dob = DateValue("Oct 30, 1966") MsgBox Format(dob, "dddd")

End Sub

**HomeWork** **Exercises** :

1.You are given a date. Find the month of that date.

eg for 17/7/2005:



MsgBox Month(Date)

2. Use DateSerial to find the date of the first day of that month of that year.



Express this as a day of the week:



* 3. More difficult: Find the date of the last day of a month.

There is no simple VBA function. You will need to construct one.

4. The code on page 176 of the book finds the number of weekdays (working days) between

Dec 1,2004 and Mar 1,2005.

Modify the book code to exclude these public holidays:

“Dec 25,2004", "Jan 1,2005", "Jan 26 ,2005".

Your result should be:



(Note that “Dec 25,2004" and "Jan 1,2005" are both Saturdays.)

Hint: Put the holidays into an array: Something like this:

stHols = Array("Dec 25,2004", "Jan 1,2005", "Jan 26 ,2005")

and iterate through the array for a match for each day i.

You may wish to also use a Boolean variable isHol and set it equal to True if a match is found.