Chapter 7

Date Routines

Links: The Chapter Name, above, moves to the next chapter. All page header links go to the contents menu. Within the chapter, Topic Titles returns here. Other Red Links are to other locations in this paper. Blue Links are to web sources.

This chapter discusses some of the many date routines possible in WordPerfect macros. It is by no means complete as to all possibilities.

! DATE SYSTEM VARIABLES. Three or four Date System Variables are available, ?DateDay, ?DateMonth, and ?DateYear, and, for WordPerfect 8 and higher, ?DateWeekDay.

- ?DateDay returns the numeric day of your computer's internal clock, 1-31. In WordPerfect 8 and higher, you can also use ?DateWeekDay to return the string value of the actual day of the week, e.g., Wednesday.
- ?DateMonth returns the numeric month of your computer's internal clock, i.e., 1 through 12.
- ?DateYear returns the numeric year of your computer's internal clock, e.g., 2003.

So, should you wish to assign variable "vDate" the date that the macro runs in a mm/dd/yyyy date format, you could do this: vDate=?DateMonth+"/"+?DateDay+"/"+?DateYear and the value of vDate would be, for example, "7/2/2003 " (content varying based upon your computer's internal clock).

! COMMONLY USED DATE COMMANDS WHEN TYPING INTO A DOCUMENT. Although *many others* are available, those you'll probably use most often are DateCode, DateText and DateFormat(string):

- DateCode. This inserts the current computer clock date into a document, and keeps that date current if the document is opened on a later date. Although it is most used when simultaneously using the Type command, it is not part of the Type command but is entered as a stand-alone command.

So, a line of code might read,

```
Type("Signed on ") DateCode Type(".")
```

- DateText. This inserts the current computer clock date into a document, and does not keep that date current if the document is opened on a later date. Although it is most used when simultaneously using the Type command, it is not part of the Type command but is entered as a stand-alone command.

So, a line of code might read,

```
Type("Signed on ") DateText Type(".")
```

- DateFormat(string). This command identifies the particular date format you want to use in conjunction with either DateCode or DateText. Maybe you want the date format you intend to write to a document to be, "Now, on this ___ day of July, 2003,"

So, two lines of code might read,

```
Type("Signed on ") DateFormat("___ day of Month, Year(4)#") DateText [or]
Type("Signed on ") DateFormat("___ day of Month, Year(4)#") DateCode
```

If you want to use "ordinals" (e.g., 1st, 2nd, 3rd, 4th, etc.) for the "day" element and you want to have the "day" match your computer's system clock, you could do this:

```
x=?DateDay Call(setOrdinal)
DateFormat (vday+" day of ,  ")
DateText // or DateCode
Quit // or Return, if in a called label
   Label(setOrdinal)
Switch(x)
   Caseof 1: vday=x+"st"
   Caseof 2: vday=x+"nd"
   Caseof 3: vday=x+"rd"
   Caseof 21: vday=x+"st"
   Caseof 22: vday=x+"nd"
   Caseof 23: vday=x+"rd"
   Caseof 31: vday=x+"st"
   Default: vday=x+"th"
   EndSwitch Return
```

! DATE COMMANDS NOT COVERED. What's been said above covers date macro commands most "common users" will want to know. But, so that you'll be aware that many other macro date commands are available (and about which you can find information in the WordPerfect Macros on-line help file), know that these additional commands are available in WordPerfect 8.0 and higher:

DateAddDays (but see Math.wcm, below)
DateAddMonths (but see Math.wcm, below)
DateAddWeeks (but see Math.wcm, below)
DateAddYears (but see Math.wcm, below)
DateAndTime (but see Math.wcm, below)
DateDay
DateDayOfYear
DateDaysInMonth
DateDaysInYear
DateIsLeapYear (but see below for Wp8 or later)
DateMonth
DateMonthName
DateOfMonthEnd
DateOfNthDay
DateOfNthWeek
DateOfNthWeekday
DatePart
DateString
DateWeekday
DateWeekdayName
DateWeekOfYear
DateYear

J. Dan Broadhead informs us that only the following are available in WordPerfect 7.0: DateAndTime, DateDay, DateMonth, DateMonthName, DateString, DateWeekday, DateWeekdayName, and DateYear, and that none of these commands are present in WordPerfect 6.1 or 6.0.

! MASSAGING DATES. Several macro means are present to "massage" date values generated by various macro commands into string values. This shows only the means that I commonly use.

• Converting Numeric Date Values To Text. Recalling that one of my primary objectives is that my macros, where possible, work in WordPerfect 6.1 and higher, and noting that if WordPerfect 6.1 and/or 7.0 are avoided from consideration I might do this differently, this is the routine I commonly use to convert numeric date values to string date values. Note: *I avoid using Dialog Date Controls – they tend not to be cross-WordPerfect macro compatible, except in WordPerfect 8.0 and higher.* Instead, the dialogs I make use Edit Box Controls requesting date values in an mm/dd/yyyy date format, e.g., "3/3/2002", with the Edit Box maximum length being 10 (that covers all reasonably possible dates). With that background, where variable "var" contains the date string value of "mm/dd/yyyy", the following converts that value to an ordinary a date string value (e.g., July 2, 1943). Somewhere in the macro, Label(DateCnv) would be called to test a user's date entry in a dialog's Edit Box:

```
Label(DateCnv) // this tests and sets the numeric parts of a m/d/yyyy date and makes string
values
Call(dateChk) If(exists(q)) Return EndIf
   // The above calls the error-trapping routine for a correct date format
x=strPos(var;"//") // this determines if an illegal pair of forward slashes have been used
```

```
If(x>0)
   q=1
   vMsg=var+" is not a legal date.  Don't use consecutive '/' marks."  setResult Return
EndIf
x=StrPos(var;"/") // If no forward slashes have been used, the date format is illegal
If(x=0)
   q=1 vMsg=var+" is not a legal date. Use a m/d/yyyy date format." setResult Return
EndIf
If(x>0) // This extracts the month portion of the date
   mo=substr(var;1;x-1) rem=substr(var;x+1;strlen(var))
   x=StrPos(rem;"/") // This extracts the day and year portion of the date
   day=substr(rem;1;x-1) year=substr(rem;x+1;strlen(rem))
   If(strlen(mo)>2 or strlen(day)>2 or strlen(year)<>4)
   // if month, day or year string lengths are wrong, this results in variable "Q" being declared
   and equaling 1
      q=1 vMsg=var+" is not a legal date. Use a m/d/yyyy date format." Return
   EndIf
EndIf
Mo=Strnum(Mo) Day=Strnum(Day) Year=strnum(Year)
   // The values are all legal; this converts the strings to numeric values
If(year<1601)
   q=1 vMsg="Dates earlier than 1/1/1601 cannot be used. Change the date" Return
EndIf
   // The routines which are used below cannot be earlier than 1/1/1601; hence this error
   routine
If(Mo=0 or Day=0 or Year=0)
   q=1
   vMsg="Don't use zeros for months, days or years. "+Mo+"/"+Day+"/"+Year+" is not a legal
   number." Return
EndIf
If(Mo>12)
   q=1
   vMsg=Mo+" is not a legal month. Use 1~12." Return
EndIf
Switch(Mo)
   Caseof 1: vMo="January"       Caseof 7: vMo="July"
   Caseof 2: vMo="February"      Caseof 8: vMo="August"
   Caseof 3: vMo="March"         Caseof 9: vMo="September"
   Caseof 4: vMo="April"         Caseof 10: vMo="October"
   Caseof 5: vMo="May"           Caseof 11: vMo="November"
   Caseof 6: vMo="June"          Caseof 12: vMo="December"
EndSwitch
vLeap=DateIsLeapYear(year) // this is used for determining what to do with February, below
Switch(vMo)
   Caseof "February":
      Switch(vLeap)
         Caseof False: If(day>28) q=1 EndIf
         Caseof True: If(day>29) q=1 EndIf
      EndSwitch
   Caseof "April": If(Day>30) q=1 EndIf
   Caseof "June": If(Day>30) q=1 EndIf
   Caseof "September": If(Day>30) q=1 EndIf
   Caseof "November": If(Day>30) q=1 EndIf
   Default: If(Day>31) q=1 EndIf
EndSwitch
If(Exists(q)) vMsg=vMo+" "+year+" does not contain "+day+" days." Return EndIf
   // above: vMsg will be used in another dialog which reports an erroneous date
```

```
var=vMo+" "+Day+", "+Year // no error resulting, var is reassigned to the resulting string value
Return
Label(dateChk) // this tests for m/d/yyyy string legal date characters
ret=IsNumberDate(var)
If(Ret) Else vMsg=var+" is not a legal date. " q=1 EndIf  Return
Function IsNumberDate (InString)
  RefString="1234567890/"
  ForNext (Count;1;StrLen(InString);1)
     If (StrPos(RefString;SubStr(InString;Count;1))=0)
        Return (False)
     EndIf
  EndFor Return (True)
EndFunc
  // This ends the date error trapping routine – if the date is erroneous, variable Q will be declared
  and will have a value of 1; otherwise, no error (Q will be undeclared) will result and the
  elements of the date string will be used to create the string date
```

Following the called Label(dateCnv), and depending on the results obtained thereby, macro flow is returned to the point of the call.

For example, in Math.wcm, one such label is this:

```
  Label(doDates) // this routine is called for "Dates", not "Numbers" or "Lawyer dates"
var=RegionGetWindowText("vMath.B1") tempD1=var
var2=RegionGetWindowText("vMath.B2") tempD2=var2
If(var="")
  q=1
  RegionSetWindowText("vMath.S6";"Nothing to compute - enter date in top box")
  RegionSetEditSelection("vMath.B1") setResult Return
EndIf
x=substr(dsel;1;1)
Switch(x)
  Caseof "D";"W";"Y";"M":
     If(tempD2="")
        RegionSetWindowText("vMath.S6";"Nothing to compute - enter date in second box")
        RegionSetEditSelection("vMath.B2") setResult Return
     EndIf
  Default:
EndSwitch
Call(dateCnv)
If(Exists(q))
  RegionSetWindowText("vMath.S6";vMsg)
  RegionSetFocus("vMath.B2")
  RegionSetEditSelection("vMath.B2") setResult Return
EndIf
FirstN=var tempd1=mo+"/"+day+"/"+year // etc. ... the call continues since "tests" were passed
```

Top of Chapter 7 Chapter 8

NOTES

NOTES