REXX Date Processing

(Computer Dating)

Chuck Meyer
Chuck Meyer Systems Inc.
10105 30th Ave. No.
Minneapolis, MN 55441
763-546-6184
cmsi@attglobal.net
Sat, 2002-03-13
WAVV, Cincinnati (Ft.Mitchell, KY)

How Do You Calculate the … ??

- **Date of last day of *mm/yy***
- **Date of tenth Saturday in *yyyy***
- **Date of Thanksgiving, *yyyy***
- **Date of Easter, *yyyy***
- **Date of Election Day, yyyy**
- **Date of *mm/dd/yy* plus|minus *nnn* days**
- **Day of week for *mm/dd/yy***
- **First weekday on|after *mm/dd/yy***
- **Last weekday on|before *mm/dd/yy***
- **4-digit year *(yyyy)* from a 2-digit year *(yy)***
- **Leap_Year_Indicator (?ly) for *yyyy* (NOT *yy*)**

Hasn't Someone Done This Before?

- **REXX BuiltIn Functions**
  - **REXX 4.01 has DATE(f1,date,f2)**
    - **Cannot convert from|to types C or J**
  - **Personal REXX has DATECONV(date,f2,f1)**
- **VMSHARE**
  - **DATECONV.EXEC**
  - **EDATE90.EXEC**
  - **DATEC.MODULE (CMS-only)**
- **Overhead**
  - **Validation & data conversion**
  - **Not always optimal coding**
  - **Virtual storage**
- **Platform portability**
  - **Differing syntax**

Credits / Bibliography

- **Open text**
  - **Communications of the ACM**
  - **Ephemeris**
  - **Astronomical Formulae for Calculators**
  - **IBM Systems Journal**
  - **Time/Universal/World Almanacs**
- **Machine-readable**
  - **VMSHARE**
  - **RXTOOLS (Univ of Penna)**
  - **HC (Historical Calendar) PC/DOS Shareware**
    - **0001-2700**
  - **Julian Calendar Deluxe, V2.02; PC/DOS Freeware**
    - **1582-3000; Copperflow Solutions**
    - **One day slow (JDN values are low)**

Pre-Julian Calendars

- **Lunar (based on month of 29.5 days)**
  - **New moon to new moon**
  - **Months typically alternate 29-30 days**
  - **Day begins at sunset**
  - **Non-integral months per year (~11 days short)**
    - **Some (Chinese) use an intercalary month**
    - **Some (Islamic) ignore it (a year = 12 months)**
- **Solar (based on a year of 360-365.24 days)**
  - **Equinox to equinox; solstice to solstice; star rising**
  - **Babylonian (???) 360 days**
  - **Egyptian (4241 or 2773 BCE) 12*30+5 = 365 days**
  - **Mayan (300 BCE) 365.24 days**
- **Solar-lunar (19-year cycle, 7 w/ 13 months)**
  - **Jewish (intercalary days AND months)**
  - **Roman (29,31,28 days plus intercalary Mercudonius)**

History of the Julian Calendar

- **46 BCE (-45)**
  - **Lunar calendar running 80 days "fast"**
  - **Julius Caesar (Sosigenes) designed a solar calendar**
  - **Ended 6 days after Winter Solstice (12/25)**
  - **New Year's Day moved from Vernal Equinox to January 1**
  - **"Year of Confusion" – 445 days long**
- **45 BCE (-44)**
  - **First "Julian" year, also first leap year**
  - **Still using Calends/Ides/Nones**
  - **Month lengths now either 30|31 days, plus February**
- **0321-0325**
  - **7-day week (orig. Mesopotamian), 7 celestial bodies**
  - **Spring Equinox has slipped to March 21**
  - **Bishops in Nicaea decreed this date frozen**
- **0532-0732**
  - **"AD" used by Dionysius Exiguus & the Venerable Bede**
  - **Venerable Bede calculated Julian year was too long, by 11 minutes and 14 seconds (1 day every 128 years)**

History of the Gregorian Calendar

- **1582**
  - **Spring Equinox now March 11 (10 days slow)**
  - **Pope Gregory XIII (w/ Luigi Lilio) establishes "NS" calendar**
  - **Thursday, Oct. 4 was followed by Friday, Oct. 15**
  - **Three leap-days dropped each 400 years**
  - **Days of month numbered sequentially**
  - **Joseph Scaliger defined a Julian Epoch (Julian Day Number)**

- **1752 (Only in England & colonies)**
  - **Julian (OS) calendar now 11 days slow**
  - **Wednesday, Sept. 2 was followed by Thursday, Sept.14**
  - **New Year's Day moved from March 25 to January 1**
  - **N.A. colonies "Gregorianized" all earlier dates**
    - **(1731-02-11 OS became 1732-02-22 NS)**
  - **England Gregorianized some earlier YEARS (not dates)**

History of the Gregorian Calendar

- **Other countries converted at different times**
  - **1698 Germany, Netherlands**
  - **1752 Great Britain**
  - **1753 Sweden**
  - **1792-1830 French Revolutionary (Decimal) calendar**
  - **1867-10-18 Alaska (Seward's Folly)**
  - **1873-1875 Japan, Egypt**
  - **1912-1917 China, Turkey**
  - **1918-02-13 Russia**
  - **1923 Greece (2800-02-29 does not exist)**
- **3200 (estimate will be a full day slow again)**
  - **Will another leap-day be bypassed??**
  - **No consensus for a solution (WHAT or WHO)**

Calendars Futures

- **2001-06-dd BillieWare problem w/ Windoze**
- **2015-01-01 PC-DOS “GETDATE” overflow**
- **2038-01-18 Unix’s TIMEVAL o/f @ 22:14:07**
- **2042-09-17 /370 TOD-clock o/f @ 23:53:46**
- **2080-01-01 PC-DOS DATE(??)**
- **2738-11-29 REXX BaseDate rolls to 7 digits**
- **2800-03-01 One days slower than Greek Cal**
- **3268-01-23 (01/01 OS) Start next Julian Epoch**

Date Formats

- **Granularity (24-hour, local day)**
- **Three major formats**
  - **Determined by NUMBER of UNITS used (1,2,3)**
  - **Can convert in 15 (or fewer) lines of REXX code**
- **Numerous minor formats**
  - **Usually just re-ordered fields**
  - **Convert in 1 line of REXX code**
- **The ambiguous *yy* problem**

Major Date Formats

- **Three Unit (Year, Month, Day-of-month)**
  - **Many, non-obvious, minor forms**
    - **S – ccyymmdd (preferred)**
    - **U – mm/dd/yy**
    - **E – dd/mm/yy**
    - **O – yy/mm/dd**
    - **N – dd_Mmm_ccyy**
    - **ccyy-mm-dd (ISO/Japanese)**
    - **dd-Mmm-ccyy**
- **Two Unit (Year, Day-of-year)**
  - **Erroneously called JULIAN**
  - **Used by MVS and DOD (labels)**
- **One Unit (Sequential Day Number)**
  - **So When is day-zero?**
    - **Day before 4713BCE (-4712) 01-01 (Julian Day Number)**
    - **Day before xx00-01-01 (REXX's Century format – NON-SAA)**
    - **0001-01-01 (REXX's BaseDate format, OS form extrapolated back)**
    - **1582-10-14 (Lilian)**
    - **1858-11-16 (Modified JDN)**
    - **First day of 1900, 1960, 1980, ...**

Zero-th Day of Month

- A concept used in some astronomical tables
- The day before the first day of any month
- Astronomical tables show the JDN of each Zero-th Day
- Add day-of-month to get JDN for that day
- This concept is NOT supported by DATE() BIF
- This concept IS supported by all attached coding
- The zero-th day is also the last day of preceding month

Definitions

- "Julian"
  - Calendar (Julius Caesar)
  - Year (365.25 days; leap-year every 4 years)
  - Epoch (period of 7980 Julian years from 4713 BCE)
  - Date (yyddd or ccyyddd) ("MVS")
  - Day Number [JDN] (TRUNC(JD+0.5))
  - Modified Julian Day Number (JD+2400000.5)
- "Style" can mean 2 different things –
  - OS=Julian Calendar; NS=Gregorian Calendar
  - The date of New Year's Day (1/1 3/1 3/25 12/25)
- Range of accuracy for calculations
  - "Accuracy" is normally the number of significant digits
  - For date-calculations, "accuracy" refers to the number of years over which the algorithm is valid

Significant Dates in Various Formats

Greg(NS) yyyymmdd	Jul(OS) yyyymmdd	DoW	JulDay nnnnnnn	REXXBase nnnnnnn	L Y	DoC *RX	DoY *RX
4713BC	-47120101	Mon	0	-1721426	Y	*	1
45BC	-00440101	Fri	1704987	-16439	Y	*	1
00001230	00010101	Sat	1721424	-2	.	*367	1
00010101	00010103	Mon	1721426	0	.	*369	3
15821014	15821004	Thu	2299160	577734	.	30228	277
15821015	15821005	Fri	2299161	577735	.	30229	278
15821231	15821221	Fri	2299238	577812	.	30306	355
17320222	173x0211	Fri	2353712	632286	Y	11740	53
17520914	17520903	Thu	2361222	639796	Y	19250	258
18000101	17991221	Wed	2378497	657071	.	1	1
18581116	18581104	Tue	2400000	678574	.	21504	320
19000101	18991220	Mon	2415021	693595	.	1	1
19800101	19791219	Tue	2444240	722814	Y	29220	1
19940609	19940527	Thu	2449513	728087	.	34493	160
19991231	19991218	Fri	2451544	730118	.	36524	365
20000101	19991219	Sat	2451545	730119	Y	1	1
20020413	20020331	Sat	2452378	730952	.	834	103
20420917	20420904	Wed	2467145	745719	.	15601	260
27001231	27001218	Mon	2707579	986153	.	365	365
30000101	29991222	Wed	2816788	1095362	.	1	1
32680123	32680101	Mon	2914695	1193269	Y	24860	23
99991231	99991020	Fri	5373484	3652058	.	36524	365

Comparison of Year Lengths

	Tropical	Egyptian	*Julian3	Julian	Gregorian
Yr len (seconds)	31556925.9747	31536000	31564800	31557600	31556952
Yr len (365\h\m\s)	5\48\45.9747	0\0\0	8\0\0	6\0\0	5\49\12
Yr len (days)	365.24219879	365.0	365.333	365.25	365.2425
Days in century	36524.219879	36500	36533.3	36525	36524.25
Years in one cycle	~3323	1	3	4	400
Leap-years / cycle	~805	0	1	1	97
Leap-years / 4000	969	0	1333.33	1000	970
Leap-years / 1000	242.25	0	333.33	250	242.5
Ann excess (days)	.	-.242199	.0911342	.0078009	.0003009
Ann excess (h\m\s)	.	-5\48\46	2\11\14	0\11\14	0\0\26
Ann excess (secs)	.	-20926	7874	674	26
Yrs to gain a day	.	-4	12	128	3323
Yrs to gain a year	.	-1460	4380	46720	1213699
Days gained / 100y	.	-25	8.3	.78	.0301
Calendar is too	.	short	long	long	long
" is said to be	.	FAST	SLOW	SLOW	SLOW
The solution is to	.	add	drop	drop	drop

REXX Coding : Ground Rules

- **All coding shown as Procedures**
- **May remove “Procedure” if unique Var names**
- **Most samples assume input in *ccyymmdd* form**
- **Modify parsing template for other input forms**
- **No data-validation is shown (do you need??)**
- **Two primitives are used in many other places**
- **Building-block concept, use as your base**
- **All coding assumes 1582 J2G conversion**

Primitive – Is It a Leap-Year?

- **Differing Techniques depending on scope of accuracy**
- **Boolian Results: 1=True 0=False**
  ("Boolian" does not preclude algebraic use)
- **Basic Coding**

```
?LY:  Procedure
  Parse Arg 1 y +4
  Return ____ ← One of the following expressions
```

- **Valid from 1901 thru 2099**

```
((y//4)=0)
```

- **Valid from 1583 thru 3199 (or is it 9999?)**

```
((y//4)=0) & (((y//100)\=0) | ((y//400)=0))
```

- **Valid from 45BCE (-44) thru 3199**

```
((y//4)=0) & (((y//100)\=0) | ((y//400)=0) | (y<1583))
```

- **Using REXX BIFs (caution, not universal)**

```
(DATE('B',y'0301','S')-DATE('B',y'0228','S')-1)
```

Primitive – *YY* to *CCYY* Conversion

- **Used primarily for human input, not file**
- **Many different techniques possible**
- **This example uses only a 2-digit year (no month or day in the algorithm)**
- **Returns input w/ 2-digit *CC* prepended**
- **Returned date is somewhere in a 100-yr window**
  - **100-yr windowing is not always appropriate**
- **First year of window defaults to current-50**
- **Optional second parameter overrides the 50**

Primitive – *YY* to *CCYY* Conversion

- **Sample CALLS with results**
  - **In 1999, YY2YYYY(00) ➔ 2000**
  - **In 2000, YY2YYYY(00) ➔ 2000**
  - **In 2000, YY2YYYY(90) ➔ 1990**
  - **In 2001, YY2YYYY(02,99) ➔ 1902**
  - **In 2001, YY2YYYY('95-Jan 31') ➔ 1995-Jan 31**

- **The coding**

```
DAT_YY2YYYY:  Procedure
  Parse Arg   1 yy +2  1 arg1 , go_back
  Parse Value DATE('S')  With cc +2 yy_cur +2
  If (go_back='*') | \DATATYPE(yy,'W') Then
    Return  cc||arg1
  go_back = WORD('50' goback, 1+DATATYPE(go_back,|'W'))
  yy_min  = (100 + yy_cur – go_back)  // 100
  Return (cc + ((go_back < yy_cur) & (yy< yy_min)) ,
               ((go_back > yy_cur) & (yy>=yy_min))) || arg1
```

Convert Between CCYYMMDD and JDN

- **Sorted to JDN**

```
DAT_S2JDN:  Procedure
  Parse Arg yr +4 mo +2 day +2  1 s
  y = yr -     (mo < 3)
  m = mo + (12*(mo < 3))
  Return  ((365.25  *(4716+y))%1) + ,
          (( 30.6001*(   1+m))%1) + day - 1524 + ,
          ((2 - (y%100) + (y%400)) * (15821004<s))
```

- **JDN to Sorted**

```
DAT_JDN2S:  Procedure
  Parse Arg   jdn , jdhow
  Parse Value jdn+.5  With z '.' -0 f
  alfa   = ((z-1867216.25)%36524.25)
  b      = z + 1524 + (1+alfa-(alfa%4)) * (z > 2299160)
  c      = (b - 122.1 ) % 365.25
  d      = (365.25 * c) % 1
  e      = (b - d     ) % 30.6001
  dd     = b - d - ((30.6001 * e) % 1) + f
  mm     = RIGHT(e - 13 + (12*(e<13.5)),2,'0')
  yy     =       c - 4715 - (mm>2.5)
  yy     = RIGHT(yy   ,MAX(4,LENGTH(yy)),' ')
  dd     = RIGHT(dd%1 ,2,'0')
  Return   yy ||mm|| dd
```

Convert Between CCYYMMDD and BaseDate

- **Sorted to B (using BIF)**

```
DAT_S2B:    Return DATE('B',ARG(1),'S')
```

- **Sorted to B (using DAT_S2JDN)**

```
DAT_S2B:   Return DAT_S2JDN(ARG(1))-1721426
```

- **B to Sorted (using BIF)**

```
DAT_B2S:    Return DATE('S',ARG(1),'B')
```

- **B to Sorted (using DAT_JDN2S)**

```
DAT_B2S:   Return DAT_JDN2S(ARG(1)+1721426)
```

- **Sorted to JDN (using BIF)**

```
DAT_S2JDN:  Return DATE('B',ARG(1),'S')+1721426
```

- **JDN to Sorted (using BIF)**

```
DAT_JDN2S:  Return DATE('S',ARG(1)-1721426,'B')
```

Convert CCYYMMDD and CCYYDDD

- **MVS-format is the 7-digit *ccyyddd***
- **Converting Sorted to MVS**

```
DAT_S2MVS:  Procedure   Parse Arg 1 yyyy +4 mm +2 dd +2 .
  Return yyyy || RIGHT(3055*(mm+2)%100-(mm+10)%13*2-91
         + (1-((yyyy//4)+3)%4+((yyyy//100)+99)%100      ,
         - ((yyyy//400)+399)%400)*(mm+10)%13+dd  ,3,'0')
```

- **Converting MVS (*yyddd* or *ccyyddd*) to Sorted**

```
DAT_MVS2SD:  Procedure
  Parse Value REVERSE(ARG(1)) With  1 j +3 y
  Parse Value REVERSE(j y)    With  y j
  If LENGTH(y) = 2  Then     y =  YY2YYYY(y)
  months = '31' (28 + LY?(y))   ,
           '31 30 31 30 31 31 30 31 30 31'
  Do m = 1 To 12 While j > WORD(months,m)
     j = j                - WORD(months,m)
     End
  Return RIGHT(y,4,0) || RIGHT(m,2,0) || RIGHT(j,2,0)
```

Convert to Day-of-Week (DOW)

- **Returns a 0 (Sunday) thru 6 (Saturday)**
- **From JDN or B (returns a 0-6)**

  (Note that both JDN#0 and BaseDate#0 occurred on a Monday, so the same algorithm will process either form)

```
DAT_JDN2DOW:  Procedure
  Return  (ARG(1) + 1) // 7
```

- **From JDN or B (returns a 0-6 or alphabetic)**

```
DAT_JDN2DOW:  Procedure
  Parse Arg  jd , type
  w = (jd + 1) // 7
  If ABBREV('NUMERIC', type, 1)  Then Return w
  Return WORD('Sunday Monday Tuesday Wednesday' ,
              'Thursday Friday Saturday',w+1)
```

- **From Sorted (ccyymmdd)**

```
DAT_S2DOW:  Procedure
  Parse Arg  1 yyyy +4 mm +2 dd +2 . , type
  f =   yyyy + (mm-14)%12
  w =   ((13*(mm+10-(mm+10)%13*12)-1)%5+dd+77        ,
        + 5 * (f - f%100*100)%4  +  f%400 - f%100*2)  //7
  If ABBREV('NUMERIC',type,1)   Then Return w
  Return  ,
    WORD('Sun Mon Tues Wednes Thurs Fri Satur',w+1)'day'
```

Month-Length / Last-Day-of-Month

- **Input requires *yyyy* and *mm***
- **Returns either *dd* or *ccyymmdd***
- **The coding --**

```
DAT_MONTH_LEN:   Procedure  /* return CCYYMMDD */
  Parse Arg 1 yyyy  +4  mm  +2
  Return                 (30          , /* 30         */
    + ((mm>7) <> (mm – ((mm%2)*2))) , /* plus 0|1   */
    - ((2-?LY(yyyy)) *  (mm=2)))      /* less 0|1|2 */

DAT_MONTH_END:   Procedure  /* return DD */
  Parse Arg 1 yyyy  +4  mm  +2
  Return yyyy || mm ||  (30          , /* 30         */
    + ((mm>7) <> (mm – ((mm%2)*2))) , /* plus 0|1   */
    - ((2-?LY(yyyy)) *  (mm=2)))      /* less 0|1|2 */
```

On-Or-After a Specified Date

- **Given: Date (ccyymmdd) and DayOfWeek (0-6)**
- **Result: Date(ccyymmdd) which is the first date on the specified DOW, which is on-or-after the supplied date. This will be 0-6 days after the specified date.**
- **If DOW is non-numeric, the result will be the first WEEKDAY (1-5). This will be 0-2 days after the specified date.**
- **The coding –**

```
DAT_ON_OR_AFTER:  Procedure
  Parse Arg dow , yyyymmdd          /* Input date is in YYYYMMDD  */
  jd1  = DAT_S2JD(yyyymmdd)         /* Input date into JulianDate */
  dow1 = (jd1+1) // 7               /*   and then its day-of-week */
  If VERIFY(dow,'0123456') > 0 Then ,  /* wants next WEEKDAY ??  */
     If (dow1>0) & (dow1<6)    Then Return yyyymmdd    /* OK     */
                               Else dow = 1     /* set to Monday */
  bump = (('0'dow)-dow1+7)//7       /* How many days needed ??    */
  Return DAT_JD2S(jd1+bump)         /* Add it; cvt to Sorted; Ret */
```

- **For DAT_ON_OR_BEFORE, replace last 3 lines with –**

```
                               Else dow = 5     /* set to Friday */
  bump = (dow1-('0'dow)+7)//7       /* How many days needed ??    */
  Return DAT_JD2S(jd1-bump)         /* Sub it; cvt to Sorted; Ret */
```

Easter

- **Given: 4-digit year (yyyy)**
- **Result: Date (ccyymmdd) of Easter in that year**
- **Different algorithms for OS and NS**
- **The coding –**

```
DAT_EASTER:  Procedure
 /* From Astronomical Formulae for Calculators;  Chap 4,  Page 31-33 */
 Parse Arg 1 cc +2 yy +2 1 yyyy +4 . , how
 If yyyy > 1582  Then Do /* GREGORIAN CAL; valid 1583 thru 9999     */
    a = yyyy                                      // 19
    g = (cc - ((cc+8) % 25) + 1)                   % 3
    h = (19*a + cc - (cc%4) - g + 15 )            // 30
    l = (32 + 2*(cc//4) + 2*(yy%4) - h - (yy//4)) // 7
    q = (h + l - 7*((a + 11*h + 22*l)%451) + 114)
    End
 Else Do                 /* JULIAN CAL;    valid 0325 thru 1582      */
      /* For years prior to 325, there was no agreement on the date  */
      /* for Easter, so these results are speculative, at best.      */
    d = (19*(yyyy//19) + 15)                    // 30
    e = ( 2*(yyyy// 4) + 4*(yyyy//7) - d + 34) // 7
    q = (d + e + 114)
    End
 Return yyyy || RIGHT(q%31,2,'0') || RIGHT((q//31)+1,2,'0')
```

Create New Output Date Formats

- **Input: Any format except ‘J’ and ‘C’**
- **Output: Any format including ‘J’, ‘C’, and ‘I’ (ISO)**
- **Syntax: DAT$(fmt_out , date , fmt_in)**
- **PreReq: REXX at 4.01 (w/ 3-arg DATE())**
- **The coding –**

```
DAT$:      Procedure /* like DATE()  -- but ALL formats handled!!  `*/
  Parse       Arg  fo    , dx , fi       /* allows “N”-format input */
  Parse Upper Arg  fo +1 , .  , fi +1    /* use first letter only   */
  Parse Value DATE('S',dx,fi )  With  ds . 1 cc +2 yy +2  1 y4 +4
  If (fo='I') Then Return  TRANSLATE('CcYy-Mm-Dd',ds,'CcYyMmDd')
  If (fo='J') Then Return  yy || RIGHT(DATE('D',ds,'S'),3,0)
  If (fo='C') Then Return  DATE('B',ds,'S') - DATE('B',cc'000101','S')
  If (fo=‘V’) Then Return TRANSLATE(DATE(‘N’,ds,’S’),’-’,’ ’)
  Return DATE(WORD(fo ‘N’,1) , dx , WORD(fi 'N',1))
```

Goodies

- From BIM's website (www:bimoyle.com/download) download WAVRXDAT.ZIP (about 130KB) and extract the components on Windoze ...
  - DATES.XLS is an Excel spreadsheet showing various dates in multiple formats
  - W@1DAT.A01 is a "read me" file (!! INSTRUCTIONS !!)
  - WAVRXDAT.PPT is this PowerPoint presentation
  - W@1DAT.A02 and W@2DAT.A03 are directories
  - W@1DAT.REX must be uploaded to VM|VSE|MVS as an ASCII,CRLF member (this is the only component of the ZIP file which is targeted for the mainframe)
    - It may be given any name on the mainframe.
    - It must eventually reside wherever REXX programs must reside on your OpSys, in order to be executed.
      - VSE: Library member xxxx.PROC
      - VM: File named xxxx.EXEC
      - MVS: Either a sequential file, or a member in any PDS which can be ALLOCated to SYSEXEC. (if RECFM=V is specified, LRECL must be at least 84)
      - OS/2: File named xxxx.CMD
      - Personal-REXX on DOS, Windoze, or OS/2: File named xxxx.REX

Contents of WAVRXDAT.REX

FileTypes will be set as required for the target OpSys

$$$INDX1	.	1	Table of Contents
$README$	REX	.	Description of CAL & HOLIDAYS
@@DAT01	REX	.	All coding samples used herein
CAL	REX	.	CAL application program
DAT@H2J	REX	.	Function: Hebrew-to-JDN
DAT@J2H	REX	.	Function: JDN-to-Hebrew
DAT@SOL	REX	.	Function: Solstice/Equinox
DAT@SOLT	REX	.	Test driver for DAT,SOL
HOLIDAYS	REX	$PR,$DA,$HR	Function: w/ 3 sample profiles
HOLIDAYT	REX	LST	Test driver & sample output
QD	REX	.	A MFC original (Query Date)

Contents of @@DAT01.REX

These subroutines are available for recycling

LY?n	Is this a leap-year?? (3 algorithms)
DAT_YY2YYYY	Convert a 2-digit-year to 4-digits
DAT_S2JDN	Sorted-Date to JulianDayNumber
DAT_JDN2S	JulianDayNumber to Sorted
DAT_S2MVS	Sorted to MVS (*ccyyddd*)
DAT_MVS2S	MVS (*ccyyddd* or *yyddd*) to Sorted
DAT_JDN2DOW	JulianDateNumber to DayOfWeek
DAT_MONTH_LEN	Number of days in the month
DAT_MONTH_END	*ccyymmdd* of last-day-of-month
DAT_ON_OR_AFTER	*ccyymmdd* for a specific DayOfWeek
DAT_ON_OR_BEFORE	*ccyymmdd* for a specific DayOfWeek
DAT_EASTER	*ccyymmdd* of Easter for specified year
DAT$	Adding functionality to DATE()

HOLIDAYS - 1

- Runs on VSE, CMS, TSO, PC (PersonalRexx)
- Must be called as a FUNCTION
  - hols = HOLIDAYS('2002')
- Needs a profile which defines each holiday
  - OpSys-specific name for profile (can be overridden)
    - CMS, GCS: HOLIDAYS $PROFILE *
    - PC,VSE: HOLIDAYS.$PR
    - TSO: HOLIDAYP
  - Attributes
    - Holiday vs. Day-name
    - Whether Date-specific, Day-specific, or other
    - Range of years that this holiday is valid
- Returns a string of all holidays separated by '00'x
- Returned holidays are in calendar sequence
  - If date collisions occur, in PROFILE sequence

HOLIDAYS - 2

- **Returns the following data for each holiday**
  - **Date in sorted (CCYYMMDD) format**
  - **JulianDayNumber (subtract 1721426 for BaseDate)**
  - **Short name of holiday (10 characters)**
  - **Long name of holiday**
  - **Type of holiday (“H”, “h”, x)**
- **Sample profiles provided**
  - **$PR is a full sample with most holidays and comments**
  - **$CH, $DA, $HR are specialized examples**
- **HOLIDAYT is a sample program**
  - **It calls HOLIDAYS for a specified year(s), shows results**
  - **May specify a different profile-name**
- **$README$.PROC shows full description**

CALendar

- Runs on VSE, CMS, TSO, PC (PersonalRexx)
- Creates a disk file that is a printable calendar
  - LRECL 80
  - CCC in column 1 (data in 2 thru 79)
  - 7 weeks per page (49 @ 10x7 cells)
  - Can specify start-date and num-of-weeks
  - VSE output defaults to libr.sublib.CAL.LISTING
    - Option “D=SYSLST” routes direct to SYSLST (w/ CC)
    - Option “D=dlbl” routes to specified DLBL
    - Option “D=l.s.fn.ft” routes to member of sublibrary
- Each cell has date in several formats
  - Holiday names from HOLIDAYS (may be suppressed)
- Program may be easily modified to alter cell contents
  - Vars “lin.2” thru “lin.6” (.7 & .8 used for HOLIDAYS)
  - Get data from other programs (viz., Shop Floor Control)

Other Goodies

- QD.PROC (Query Date)
  - Written by MFC in early 80's
  - Displays today's (or specified) date in multiple formats
  - Also displays the lunar phase (cute)
  - Reflects OS information before 1582-10-15

- @@DAT01.PROC
  - Contains all coding used in this slide presentation
  - It's execution will exercise most of the code
  - One of these formats
    - @@DAT01
    - @@DAT01 ccyymmdd
    - @@DAT01 yymmdd

DATE's Over !!