Subject: Re: Built-in date / time routines
Posted by davidf on Tue, 18 Apr 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Mark Hadfield (m.hadfield@niwa.cri.nz) writes:

> Oh, and complain to RSI!

Oh, man, talk about adding fuel to the fire...

It took me about 6 months of constant effort to
get Andy calmed down from the Contour plot stuff. :-(

Cheers,

David

P.S. Let's just say the good folks at RSI are probably
already familiar with Andy Loughe's name.

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Built-in date / time routines
Posted by davidf on Tue, 18 Apr 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Andy Loughe (loughe@fsl.noaa.gov) writes:

> I have inherited code which utilizes these built-in IDL routines:
> var_to_dt
> dt_subtract
> dt_to_var
> dt_add
>
> Would someone please tell me what happened to these?
> This time I could not even find them in the obsolete directory.

These routines flared into brilliance and then disappeared
like one of those...uh, stars. (My God, I've just alienated
the entire Astronomy community with my poor memory.)

Apparently someone's great idea was not Y2K compliant. :-(

I think they can be found in only one version of IDL
(the one your friends were running, apparently).

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Built-in date / time routines
Posted by Andy Loughe on Wed, 19 Apr 2000 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> Mark Hadfield (m.hadfield@niwa.cri.nz) writes:
>
>> Oh, and complain to RSI!
>
> Oh, man, talk about adding fuel to the fire...
>
> It took me about 6 months of constant effort to
> get Andy calmed down from the Contour plot stuff. :-(
>
> Cheers,
>
> David
>
> P.S. Let's just say the good folks at RSI are probably
> already familiar with Andy Loughe's name.

My last great bone of contention was the beta-test agreement
that would have caused conflict with my employer. Unfortunately,
the deletion of !map.out came at the same time.
(Lots of good code had to be re-written in those days).

A kind soul at RSI took me out to lunch and helped me see some
of what they are up against. For that kindness, I am grateful.

It doesn't mean I always understand their decisions, but it has made me
more tolerant of actions which affect my effectiveness as a programmer.

Subject: Re: Built-in date / time routines
Posted by wmc on Wed, 19 Apr 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Mark Hadfield <m.hadfield@niwa.cri.nz> wrote:
> "Andy Loughe" <loughe@fsl.noaa.gov> wrote in message
> news:38FC79DC.41131032@fsl.noaa.gov...
>>
>> I have inherited code which utilizes these built-in IDL routines:
>> var_to_dt
>> dt_subtract
>> dt_to_var
>> dt_add
>>
>> Would someone please tell me what happened to these?

> The routines were all implemented in code so if you can find a copy of them
> they should still work fine...

Well, as a service to the world, here they are (from, I think, 5.2 sources).
I think you need idldt__define.pro too. ls *dt* in 5.2 gives:

/tmp_mnt/nerc/packages/idl/v5.2/idl_5.2/lib/compat/dt_add.pr o
/tmp_mnt/nerc/packages/idl/v5.2/idl_5.2/lib/compat/dt_plot.p ro
/tmp_mnt/nerc/packages/idl/v5.2/idl_5.2/lib/compat/dt_subtra ct.pro
/tmp_mnt/nerc/packages/idl/v5.2/idl_5.2/lib/compat/dt_to_sec .pro
/tmp_mnt/nerc/packages/idl/v5.2/idl_5.2/lib/compat/dt_to_str .pro
/tmp_mnt/nerc/packages/idl/v5.2/idl_5.2/lib/compat/dt_to_var .pro
/tmp_mnt/nerc/packages/idl/v5.2/idl_5.2/lib/compat/dtgen.pro
/tmp_mnt/nerc/packages/idl/v5.2/idl_5.2/lib/compat/idldt__de fine.pro
/tmp_mnt/nerc/packages/idl/v5.2/idl_5.2/lib/compat/jul_to_dt .pro
/tmp_mnt/nerc/packages/idl/v5.2/idl_5.2/lib/compat/sec_to_dt .pro
/tmp_mnt/nerc/packages/idl/v5.2/idl_5.2/lib/compat/str_to_dt .pro
/tmp_mnt/nerc/packages/idl/v5.2/idl_5.2/lib/compat/var_to_dt .pro

No guarantee that I've found the right files or that they work...
In fact, looking closer, I see that I thought var_to_dt needed some patching:
I include my version too.

-W.

```
;+
; NAME:
;     VAR_TO_DT (***kludged by WMC***)
;
;
; PURPOSE:
;     Convert values representing date and time to an IDLDT date/time
;     variable.
```

```
;
;
; CATEGORY:
;      Misc
;
;
; CALLING SEQUENCE:
;      dtvar = VAR_TO_DT( yyyy, mm, dd, hh, min, ss)
;
;
; INPUTS:
;
; OPTIONAL INPUTS:
;      yyyy: A scalar or array containing integer year(s).
;      mm:   A scalar or array containing integer month(s)
;      dd:   A scalar or array containing integer day(s) of the month.
;      hh:   A scalar or array containing integer hour(s) of the day.
;      min:  A scalar or array containing integer minute(s).
;      ss:   A scalar or array containing the float seconds.
;
;
; KEYWORD PARAMETERS:
;
;
;
;
; OUTPUTS:
;      VAR_TO_DT returns the IDLDT date/time structure containing the
;      converted date(s).
;
;
;
;
; OPTIONAL OUTPUTS:
;
;
;
;
; COMMON BLOCKS:
;      NONE
;
;
; SIDE EFFECTS:
;      The result is a named IDLDT date/time structure.  If the
;      structure named IDLDT has not been defined, IDL invokes the
;      IDLDT__DEFINE procedure to create it.  IDLDT__DEFINE creates
;      a number of system variables.
;
;
;
;
; RESTRICTIONS:
```

```
;     If any of the inputs are arrays, all of the inputs, if present,
;     must be arrays of the same dimension.
;
;
;
; PROCEDURE:
;
;
; EXAMPLES:
;     date = VAR_TO_DT( 1997, 12, 21 ) ; Create an IDLDT with default
;                                      ; Values for hh, mm, ss.
;     PRINT, date
;
; The result is
;     { 1997 12 21 0 0.00000 0.0000000 0 }
;
;
;     date = VAR_TO_DT( [1997, 1998], [12, 1], [1, 1] )
;
; MODIFICATION HISTORY:
;
;-

FUNCTION VAR_TO_DT, yyyy, mm, dd, hh, min, ss

nElements = n_elements( yyyy )

if nElements eq 0 then begin

   retVal = !dt_base
; <WMC> Fails if yyyy is an array with 1 element!
; endif else if nElements eq 1 then begin
endif else if not isarray(yyyy) then begin
; </WMC>
   retVal = {IDLDT}
   ; Defaults
   if n_elements( mm ) eq 0 then mm = 1B
   if n_elements( dd ) eq 0 then dd = 1B
   if n_elements( hh ) eq 0 then hh = 0B
   if n_elements( min ) eq 0 then min = 0B
   if n_elements( ss ) eq 0 then ss = 0.
   retVal = {IDLDT, yyyy, mm<12>1, dd<31>1, hh<23>0, min<59>0, $
          ss<60>0, 0.d, 0b}
   retVal.Julian = julday( mm<12>1, dd<31>1, yyyy, hh<23>0, min<59>0, $
                     ss<60>0 )
endif else begin
   ; Defaults
   if n_elements( mm ) eq 0 then mm = replicate( 1B, nElements )
```

```
   if n_elements( dd ) eq 0 then dd = replicate( 1B, nElements )
   if n_elements( hh ) eq 0 then hh = replicate( 0B, nElements )
   if n_elements( min ) eq 0 then min = replicate( 0B, nElements )
   if n_elements( ss ) eq 0 then ss = replicate( 0., nelements )
   retVal = replicate( {IDLDT}, nElements )
   for i = 0, nElements-1 do begin
      retVal[i] = VAR_TO_DT( yyyy[i], mm[i], dd[i], $
                             hh[i], min[i], ss[i] )
   endfor
endelse
return, retVal
end

; $Id: idldt__define.pro,v 1.3 1998/08/19 17:14:49 griz Exp $
;
; Copyright (c) 1997-1998, Research Systems, Inc.  All rights reserved.
;       Unauthorized reproduction prohibited.
;+
; NAME:
;           IDLDT__DEFINE
;
;
; PURPOSE:
;         Automatic definition procedure for the IDLDT date/time structure
;
;
; COMMON BLOCKS:
;
;
; SIDE EFFECTS:
;         Creates the named structure, IDLDT, and a number of system
;         variables.
;
; MODIFICATION HISTORY:
;
;     Dec 1997, Dr. G. Scott Lett, RSI, Written
;
;
;-
PRO IDLDT__DEFINE


tmp = {IDLDT, Year:0, Month:0B, Day:0B, $
       Hour:0B, Minute:0B, Second:0., $
       Julian:0.D, $
       Recalc:0B $
       }
```

```
; Initialize Date/Time system variables

if n_elements( idldt_date_base ) eq 0 then begin
   DEFSYSV, '!dt_base', {IDLDT, -4713, 1, 1, 12, 0, 0, 0.d, 0}
   DEFSYSV, '!date_separator', '/'
   DEFSYSV, '!time_separator', ':'
   DEFSYSV, '!holidays', replicate( {IDLDT}, 50 )
   DEFSYSV, '!weekend_list', replicate( 0B, 7 )
   DEFSYSV, '!day_names', ['Sunday', 'Monday', 'Tuesday', $
                    'Wednesday', 'Thursday', 'Friday', 'Saturday' ]
   DEFSYSV, '!month_names',['January','February','March','April','May', $
                         'June', 'July', 'August', 'September', $
                         'October', 'November', 'December' ]
endif

END

; $Id: dt_add.pro,v 1.5 1998/08/19 17:14:45 griz Exp $
;
; Copyright (c) 1997-1998, Research Systems, Inc.  All rights reserved.
;       Unauthorized reproduction prohibited.

;+
; NAME:
;   DT_ADD
;
;
; PURPOSE:
;   Increments an array of IDLDT date/time variables by a constant amount.
;
;
; CATEGORY:
;
;
;
; CALLING SEQUENCE:
;   result = DT_ADD( dt_var )
;
;
; INPUTS:
;   dt_var: An array of IDLDT date/time structures
;
;
; OPTIONAL INPUTS:
;
;
;
; KEYWORD PARAMETERS:
```

```
;    Day: The offset in days
;    Hour: The offset in hours
;    Minute: The offset in minutes
;    Second: The offset in seconds
;    Year: The offset in years
;
;
; OUTPUTS:
;    The result is an array of IDLDT date/time structures, incremented
;    by the desired amount
;
; COMMON BLOCKS:
; None.
; SIDE EFFECTS:
; None.
; RESTRICTIONS:
; Year, Month, Days must be integers for correct operation.
; DT_ADD and DT_SUBTRACT are not necessarily commutative when
; the YEAR or MONTH keyword are used because these time units do
; not have a constant number of days.  For example, ((Jan 31 + 1
; Month) - 1 Month) is NOT Jan 31.  One month
; after January 31st is March 3rd, unless its a leap year.
; One month before March 3rd is February 3rd.
;
;
; PROCEDURE:
;    Each value is added to the corresponding component, then the
;    result is renormalized with JUL_TO_DT.
;
;
; EXAMPLE:
;    print, DT_ADD( TODAY(), day=1, year=3) ; Adds 1 to today's day
;                                           ; and 3 to today's year.
;
;
; MODIFICATION HISTORY:
;    Dec 1997, Dr. G. Scott Lett, RSI, Written
;    Jun 1998, DMS, Corrected.
;-
FUNCTION DT_ADD, dt_var, Day=day, Hour=hour, $
         Minute=minute, Month=month, Second=second, Year=year, $
         Round=iround

tyear = keyword_set(year) ? long(year) : 0
tmonth = keyword_set(month) ? long(month) : 0
tday = keyword_set(day) ? day : 0
thour = keyword_set(hour) ? hour : 0
tminute = keyword_set(minute) ? minute : 0
```

```
tsecond = keyword_set(second) ? second : 0

; Change in time, in days and fractions.
delta = tday + (thour / 24.0d0) + (tminute/1440.0d0) + (tsecond / 86400.0d0)

if abs(tmonth) gt 12 then begin ;Reduce month modulo 12
   tyear = tyear + long(tmonth)/12
   tmonth = tmonth mod 12
endif

retVal = dt_var
FOR i = 0L, n_elements(dt_var)-1 DO BEGIN

   ;; First do calculations based in variable length units,
   ;; e.g. those other than days & secs

   if dt_var[i].recalc then $
     dt_var[i] = VAR_TO_DT(dt_var[i].year, dt_var[i].month, dt_var[i].day, $
                           dt_var[i].hour, dt_var[i].minute, dt_var[i].second)

   if tyear ne 0 or tmonth ne 0 then begin
      Nyear = dt_var[i].year + tyear
      Nmonth = dt_var[i].month + tmonth
      if Nmonth lt 1 then begin
         Nmonth = Nmonth + 12
         Nyear = Nyear - 1
      endif else if Nmonth gt 12 then begin
         Nmonth = Nmonth - 12
         Nyear = Nyear + 1
      endif
; Because adding months and years can cause an invalid date, start at
; the first of the month.
      julian = julday(Nmonth, 1, Nyear, $ ;Always use 1st of month
                      dt_var[i].Hour, dt_var[i].Minute, dt_var[i].Second)
      julian = julian + (dt_var[i].day-1) ;Then add day....
   endif else julian = dt_var[i].julian

   julian = julian + delta

   if keyword_set(iround) then begin
      days = long(julian)
      julian = days + Round((julian - days) * 86400d0) / 86400d0
   endif

   retVal[i] = JUL_TO_DT(julian)
ENDFOR
RETURN, retVal
END
```

```
; $Id: dt_to_var.pro,v 1.4 1998/08/19 17:14:47 griz Exp $
;
; Copyright (c) 1997-1998, Research Systems, Inc.  All rights reserved.
;       Unauthorized reproduction prohibited.
;+
; NAME:
;     DT_TO_VAR
;
;
; PURPOSE:
;     Converts IDLDT date/time variables into numerical data.
;
;
; CATEGORY:
;
;
;
; CALLING SEQUENCE:
;     DT_TO_VAR, dt_dates
;
;
; INPUTS:
;     dt_dates:  A scalar or array of IDLDT date/time structures.
;
;
; OPTIONAL INPUTS:
;     NONE
;
;
; KEYWORD PARAMETERS:
;     Year: Integer variable to contain years
;     Month: Byte variable to contain months.
;     Day: Byte variable to contain days of the month.
;     Hour: Byte veriable to contain hours of the day.
;     Minute: Byte variable to contain minutes.
;     Second: Float variable to contain seconds.
;
;
; OUTPUTS:
;     NONE
;
;
; OPTIONAL OUTPUTS:
;     See the keywords
;
;
; COMMON BLOCKS:
```

```
;     NONE
;
;
; SIDE EFFECTS:
;     None
;
;
; RESTRICTIONS:
;
;
;
; PROCEDURE:
;
;
;
; EXAMPLE:
;     DT_TO_VAR, TODAY(), Year=years, Day=days, Month=month
;
;
; MODIFICATION HISTORY:
;
;       Dec 1997, Dr. G. Scott Lett
;
;
;
;-
PRO DT_TO_VAR, dt_days, YEAR=years, MONTH=months, DAY=days, HOUR=hours, $
              MINUTE=minutes, SECOND=seconds

IF n_elements( dt_days) GT 0 THEN BEGIN
   IF ARG_PRESENT( years ) THEN years = dt_days.year
   IF ARG_PRESENT( months ) THEN months = dt_days.month
   IF ARG_PRESENT( days ) THEN days = dt_days.day
   IF ARG_PRESENT( hours ) THEN hours = dt_days.hour
   IF ARG_PRESENT( minutes ) THEN minutes = dt_days.minute
   IF ARG_PRESENT( seconds ) THEN seconds = dt_days.second
ENDIF
END

; $Id: dt_subtract.pro,v 1.6 1998/10/02 17:27:56 griz Exp $
;
; Copyright (c) 1997-1998, Research Systems, Inc.  All rights reserved.
;       Unauthorized reproduction prohibited.

;+
; NAME:
;   DT_SUBTRACT
;
```

```
;
; PURPOSE:
;   Decrements an array of IDLDT date/time variables by a constant amount.
;
;
;
; CATEGORY:
;
;
;
;
; CALLING SEQUENCE:
;   result = DT_SUBTRACT( dt_var )
;
;
; INPUTS:
;   dt_var: An array of IDLDT date/time structures
;
;
; OPTIONAL INPUTS:
;
;
;
;
; KEYWORD PARAMETERS:
;    Compress: Eliminate holidays
;    Day: The offset in days
;    Hour: The offset in hours
;    Minute: The offset in minutes
;    Second: The offset in seconds
;    Month: The offset in months.
;    Year: The offset in years
;
;
;
; OUTPUTS:
;    The result is an array of IDLDT date/time structures, decremented
;    by the desired amount
;
; COMMON BLOCKS:
; None.
; SIDE EFFECTS:
; None.
; RESTRICTIONS:
; Year, Month, Days must be integers for correct operation.
;
; DT_ADD and DT_SUBTRACT are not necessarily commutative when
; the YEAR or MONTH keyword are used because these time units do
; not have a constant number of days.  For example, ((Jan 31 + 1
; Month) - 1 Month) is NOT Jan 31.  One month
; after January 31st is March 3rd, unless its a leap year.
; One month before March 3rd is February 3rd.
```

```
;
; PROCEDURE:
;    Each value is added to the corresponding component, then the
;    result is renormalized with JUL_TO_DT.
;
; PROCEDURE:
;    Each value is subtracted from the corresponding component, then the
;    result is renormalized with JUL_TO_DT.
;
;
; EXAMPLE:
;    print, DT_SUBTRACT( TODAY(), day=1, year=3) ; Takes 1 from today's day
;                                                ; and 3 from today's year.
;
;
; MODIFICATION HISTORY:
;    Dec 1997, Dr. G. Scott Lett, RSI, Written
;    Jun 1998, DMS, Corrected.
;-
FUNCTION DT_SUBTRACT, dt_var, Compress=compress, Day=day, Hour=hour, $
          Minute=minute, Month=month, Second=second, Year=year

tyear = keyword_set(year) ? year : 0
tmonth = keyword_set(month) ? month : 0
tday = keyword_set(day) ? day : 0
thour = keyword_set(hour) ? hour : 0
tminute = keyword_set(minute) ? minute : 0
tsecond = keyword_set(second) ? second : 0

; Change in time, in days and fractions.
delta = tday + (thour / 24.0d0) + (tminute/1440.0d0) + (tsecond / 86400.0d0)

if abs(tmonth) gt 12 then begin ;Reduce month modulo 12
   tyear = tyear + long(tmonth)/12
   tmonth = tmonth mod 12
endif

retVal = dt_var
FOR i = 0L, n_elements(dt_var)-1 DO BEGIN

   ;; First do calculations based in variable length units,
   ;; e.g. those other than days & secs

   if dt_var[i].recalc then $
     dt_var[i] = VAR_TO_DT(dt_var[i].year, dt_var[i].month, dt_var[i].day, $
                           dt_var[i].hour, dt_var[i].minute, dt_var[i].second)

; This must be reverse process of dt_add.  Otherwise, we get errors
```

```
; involving leap-years and February.

   if delta ne 0 then begin
      tmp = JUL_TO_DT(dt_var[i].julian - delta)
   endif else tmp = dt_var[i]

   if tyear ne 0 or tmonth ne 0 then begin
      Nyear = tmp.year - tyear
      Nmonth = tmp.month - tmonth
      if Nmonth lt 1 then begin
         Nmonth = Nmonth + 12
         Nyear = Nyear - 1
      endif else if Nmonth gt 12 then begin
         Nmonth = Nmonth - 12
         Nyear = Nyear + 1
      endif
      tmp = VAR_TO_DT(Nyear, Nmonth, tmp.day, $
                  Tmp.hour, Tmp.minute, Tmp.second)
   endif
   retVal[i] = tmp
ENDFOR
RETURN, retVal
END

; $Id: var_to_dt.pro,v 1.5 1998/08/19 17:14:53 griz Exp $
;
; Copyright (c) 1997-1998, Research Systems, Inc.  All rights reserved.
;       Unauthorized reproduction prohibited.

;+
; NAME:
;      VAR_TO_DT
;
;
; PURPOSE:
;      Convert values representing date and time to an IDLDT date/time
;      variable.
;
;
; CATEGORY:
;      Misc
;
;
; CALLING SEQUENCE:
;      dtvar = VAR_TO_DT( yyyy, mm, dd, hh, min, ss)
;
;
; INPUTS:
```

```
;
; OPTIONAL INPUTS:
;       yyyy: A scalar or array containing integer year(s).
;       mm:   A scalar or array containing integer month(s)
;       dd:   A scalar or array containing integer day(s) of the month.
;       hh:   A scalar or array containing integer hour(s) of the day.
;       min:  A scalar or array containing integer minute(s).
;       ss:   A scalar or array containing the float seconds.
;
;
; KEYWORD PARAMETERS:
;
;
;
; OUTPUTS:
;       VAR_TO_DT returns the IDLDT date/time structure containing the
;       converted date(s).
;
;
;
; OPTIONAL OUTPUTS:
;
;
;
; COMMON BLOCKS:
;       NONE
;
;
; SIDE EFFECTS:
;       The result is a named IDLDT date/time structure.  If the
;       structure named IDLDT has not been defined, IDL invokes the
;       IDLDT__DEFINE procedure to create it.  IDLDT__DEFINE creates
;       a number of system variables.
;
;
;
; RESTRICTIONS:
;       If any of the inputs are arrays, all of the inputs, if present,
;       must be arrays of the same dimension.
;
;
;
; PROCEDURE:
;
;
;
; EXAMPLES:
;       date = VAR_TO_DT( 1997, 12, 21 ) ; Create an IDLDT with default
;                                        ; Values for hh, mm, ss.
```

```
;      PRINT, date
;
; The result is
;      { 1997 12 21 0 0.00000 0.0000000 0 }
;
;
;      date = VAR_TO_DT( [1997, 1998], [12, 1], [1, 1] )
;
; MODIFICATION HISTORY:
; DMS, June, 1998, Cleaned-up logic.
;
;-

FUNCTION VAR_TO_DT, yyyy, mm, dd, hh, min, ss

nElements = n_elements( yyyy )

if nElements eq 0 then begin
   dummy = {IDLDT}            ;Define date structs
   retVal = !dt_base
endif else begin
   retVal = replicate({IDLDT}, nElements)
   for i=0L, nElements-1 do begin
      retVal[i] = { IDLDT, yyyy[i], $ ;Supply defaults if necessary
                    (n_elements(mm) gt i ? mm[i] : 1) < 12 > 1, $
                    (n_elements(dd) gt i ? dd[i] : 1) < 31 > 1, $
                    (n_elements(hh) gt i ? hh[i] : 0) < 23 > 0, $
                    (n_elements(min) gt i ? min[i] : 0) < 59 > 0, $
                    (n_elements(ss) gt i ? ss[i] : 0) < 60 > 0, $
                    0.d, 0b}
                          ;Obtain Julian date
      retVal[i].Julian = julday( retVal[i].month, retVal[i].day, $
                                 retVal[i].year, retVal[i].hour, $
                                 retVal[i].minute, retVal[i].second)
   endfor
endelse
return, retVal
end
```

--
William M Connolley | wmc@bas.ac.uk | http://www.nbs.ac.uk/icd/wmc/
Climate Modeller, British Antarctic Survey | Disclaimer: I speak for myself
(yes, BAS has at alast got rid of that irritating "public" in the URL)

Subject: Re: Built-in date / time routines
Posted by Mark Hadfield on Wed, 19 Apr 2000 07:00:00 GMT

"Andy Loughe" <loughe@fsl.noaa.gov> wrote in message
news:38FC79DC.41131032@fsl.noaa.gov...
>
> I have inherited code which utilizes these built-in IDL routines:
> var_to_dt
> dt_subtract
> dt_to_var
> dt_add
>
> Would someone please tell me what happened to these?

Don't start me on this again!

All the "DT" routines (which, incidentally, were introduced for
compatibility with PV-Wave) were deemed to be non-Y2K compliant (remember
Y2K?) because one of them (str_to_dt) made a simple-minded assumption about
2-digit years. In order to claim complete Y2K compliance for the product,
RSI yanked the whole lot in version 5.2.1.

I corresponded about this with technical people at RSI--the answer was
"Sorry, but it's a done deal"

The routines were all implemented in code so if you can find a copy of them
they should still work fine. Of course, I wouldn't dream of suggesting that
you do anything to violate RSI's copyright.

Frankly, the "DT" routines weren't all that great and it shouldn't be too
hard to replace them with something else. Check out
http://fermi.jhuapl.edu/s1r/idl/s1rlib/time/time.html

Oh, and complain to RSI!

Mark Hadfield
m.hadfield@niwa.cri.nz http://katipo.niwa.cri.nz/~hadfield/
National Institute for Water and Atmospheric Research
PO Box 14-901, Wellington, New Zealand