DALLAS SEMICONDUCTOR

# Application 31
## Algorithm For Converting Binary Seconds Into Data/Time

The DS1602/DS1603 stores the elapsed time and real-time as a binary count of seconds. This count always represents the number of seconds that have elapsed since a specified date/time origin. One popular origin is \{00:00:00\} JAN 1, 1970. Software is used to convert the date/time into seconds (for setting the time) and to convert the seconds count into a date/time (for reading the time). The software algorithm that does this is based on one particular origin and is transparent to the user setting or reading the clock.

The algorithm presented below assumes an origin of 00:00:00 JAN 1, 1970. Any year evenly divisible by four is assumed to be a leap year. This will hold true until the year 2100, which is not a leap year.

Assume the existence of the following integer variables:

| User Date/Time | | Clock Date/Time |
|---|---|---|
| sec = seconds (0 - 4.29E9) | (0-60) | x = seconds since origin |
| min = minutes | (0-60) | |
| hrs = hour | (0-23) | |
| day = date | (0-31) | |
| mon = month | (0-12) | |
| yrs = year | (0-2099) | |

In addition to the above variables, the following data array is generated. It represents the number of days elapsed in one non-leap year at the beginning of each month. The existence of this array greatly simplifies the algorithm.

| | |
|---|---|
| DMonth(1)= 0 | DMonth(7)= 181 |
| DMonth(2)= 31 | DMonth(8)= 212 |
| DMonth(3)= 59 | DMonth(9)= 243 |
| DMonth(4)= 90 | DMonth(10)=273 |
| DMonth(5)=120 | DMonth(11)=304 |
| DMonth(6)=151 | DMonth(12)=334 |
| | DMonth(13)=365 |

Pseudo-language Algorithm for setting the time:

```
iday = 365 * (yrs - 1970) + DMonth(mon)+ (day - 1)\      ; reg. days since 1/1/70
iday = iday + (yrs - 1969)/4                             ; + leap days since 1/1/70
if ((mon > 2) and ((yrs mod 4) eq 0)) then               ; if leap year and past feb
     iday = iday + 1                                     ; add this year's leap day
     endif                                               ;
  x = sec + 60 * (min + 60 * (hrs + 24 * iday))          ; compute seconds since '70
```

Pseudo-language Algorithm for reading the time:

NOTE : all divisions are assumed to result in integer values (i.e. are truncated).

```
imin := x / 60                                  ; whole minutes since 1/1/70
sec  := x - (60 * imin)                         ; leftover seconds
ihrs := imin / 60                               ; whole hours since 1/1/70
min  := imin - 60 * ihrs                        ; leftover minutes
iday := ihrs / 24                               ; whole days since 1/1/70
hrs  := ihrs - 24 * iday                        ; leftover hours
iday := iday + 365 + 366                        ; whole days since 1/1/68
lday := iday / (( 4* 365) + 1)                  ; quadyr = 4 yr period = 1461 days
qday := iday mod (( 4 * 365) + 1)               ; days since current quadyr began
if (( qday >=  (31 + 29) )) then                ; if past feb 29 then
lday := lday + 1                                ; add this quadyr's leap day to the
endif                                           ; # of quadyrs (leap days) since 68
iyrs := (iday - lday) / 365                     ; whole years since 1968
jday := iday - (iyrs * 365) - lday              ; days since 1 /1 of current year.
if (( qday <= 365 and qday >= 60 )) then        ; if past 2/29 and a leap year then
jday := jday + 1                                ; add a leap day to the # of whole
endif                                           ; days since 1/1 of current year
yrs  := iyrs + 1968                             ; compute year
mon  := 13                                      ; estimate month ( +1)
mday := 366                                     ; max days since 1/1 is 365
while  ( ( jday  <  mday  ) )                   ; mday = # of days passed from 1/1
do                                              ; until first day of current month
mon := mon - 1                                  ; mon = month (estimated)
mday := DMonth(mon)                             ; # elapsed days at first of "mon"
if ((mon > 2) and (yrs mod 4) = 0 ))            ; if past 2/29 and leap year then
mday := mday + 1                                ; add leap day
endif                                           : compute month by decrementing
enddo                                           ; month until found
day := jday - mday + 1                          ; compute day of month
```

## PASCAL PROCEDURES FOR DATE CONVERSION

The following Pascal code converts the number of elapsed days (I) since 1/1/70 into a standard date format (MO/DA/YR) and then back into days (N).

```
Const
    DM : Array[1..13] of Word =
    (0,31,59,90,120,151,181,212,243,273,304,334,365);

Procedure DAY2DATE(I: LongInt; Var YR, MO, DA: LongInt);

{converts # of elapsed days (I) to date format\}

Var
    J, N : Longint;
```

```
Begin
     N  := (I + 731) shl 2;
     YR := N div 1461;
     N  := (N - 1461 * YR) shr 2;
     MO := 1 + (N + 1) div 30;
     J  := DM[MO] + Byte((MO > 2) and (YR and 3 = 0));
     If N < J then Begin
      Dec(MO)
       J := DM[MO] + Byte((MO > 2) and (YR and 3 = 0));
     End;
     DA := N - J + 1;
     YR := YR + 1968;
 End;

Procedure DATE2DAY(YR, MO, DA: LongInt; Var N: LongInt);

{---------------------- convert date back to elapsed days (N) ------------------}

Begin
     N  := 365 * (YR - 1970) + (YR - 1969) shr 2 + DM[MO]
     + DA + Byte((MO > 2) and (YR and 3 = 0)) - 1;
End;
```