BinClock

A demo illustrating the use of graphicboxes and a bitmap to represent a binary clock.

- This demo was inspired by a very impressive looking demo by Mike Bradbury, as featured in an article in issue 143 of the Liberty BASIC Newsletter. In the article, Mike does an excellent job explaining the usage of drawing a single bitmap with or without an offset repeatedly within a single graphicbox to simulate the switching of states of LEDs in order to represent binary output. This demo is very similar, except that it uses six graphicboxes oriented "vertically" (for each digit of hours, minutes, and seconds in a digital clock) instead of one horizontal, and the offset is along the x-axis. Save the image below into LB's "bmp" subdirectory as "blueLED.bmp", then copy and paste the code below into a new .BAS file. The program uses a timer to check at a specified interval (in milliseconds) whether the current time is different from the last displayed time and makes modifications as necessary, only redrawing the seconds, minutes, and/or hours if they've changed. Interestingly, this clock is more accurate than Windows XP's time picker; on both my test machines, BinClock showed an updated time a split second before XP did! Time is shown in 24-hour format, although the code could be easily modified to implement a 12-hour AM/PM format.

---

blueLED.bmp

greenLED2.bmp

Update: Mike Bradbury has given me permission to use the LED image he created. This LED looks more realistic, in my opinion. Thanks, Mike! In the code below, modify the 'led$' variable to reflect the name of this bitmap, and change 'BackgroundColor$' to "lightgray".

```
nomainwin

global led$, s$, xOn, xOff

s$                  = "ff n"
p$                  = "h m s"

led$                = "blueLED.bmp"
ledFile$            = "bmp\" + led$
BackgroundColor$    = "black"

xOn                 =   0
xOff                = -15
REFRESH.INTERVAL    = 250

loadbmp led$, ledFile$
```

```
WindowWidth         = 164
WindowHeight        = 120

graphicbox #binClock.gbh10,  10, 44, 15, 34
graphicbox #binClock.gbh1,   30, 10, 15, 68
graphicbox #binClock.gbm10,  60, 27, 15, 51
graphicbox #binClock.gbm1,   80, 10, 15, 68
graphicbox #binClock.gbs10, 110, 27, 15, 51
graphicbox #binClock.gbs1,  130, 10, 15, 68

stylebits  #binClock.gbh10, 0, _WS_BORDER, 0, 0
stylebits  #binClock.gbh1,  0, _WS_BORDER, 0, 0
stylebits  #binClock.gbm10, 0, _WS_BORDER, 0, 0
stylebits  #binClock.gbm1,  0, _WS_BORDER, 0, 0
stylebits  #binClock.gbs10, 0, _WS_BORDER, 0, 0
stylebits  #binClock.gbs1,  0, _WS_BORDER, 0, 0
stylebits  #binClock,       0, 0, _WS_EX_TOPMOST, 0

open "BinClock" for window_nf as #binClock
#binClock "trapclose [exit]"

for i = 1 to 6
    ext$ = word$(p$,int((i/2) + .5)) + "1" + word$("0",(i mod 2))
    hGB$ = "#binClock.gb" + ext$
    #hGB$ "down; fill "; BackgroundColor$
next i

[refreshTime]
    timer 0
    currTime$ = time$()
    if (currTime$ <> lastDisplayed$) then
        for i = 3 to 1 step -1
            t$ = word$(currTime$,i,":")
            pT$ = word$(lastDisplayed$,i,":")
            if (t$ <> pT$) then call UpdateGBPair word$(p$,i), val(t$)
        next i
        lastDisplayed$ = currTime$
    end if

    timer REFRESH.INTERVAL, [refreshTime]
    wait

[exit]
    close #binClock
    unloadbmp led$
end
```

```
sub UpdateGBPair group$, value
    gbO$    = "#binClock.gb" + group$ + "1"
    gbT$    = gbO$ + "0"
    sO$     = upper$(group$) + "1"
    sT$     = sO$ + "0"
    tCount  = (group$ <> "h") + 1
    tens    = int(value / 10)
    ones    = value mod 10

    #gbT$   "delsegment currSegment"; sT$
    #gbO$   "delsegment currSegment"; sO$

    for i = tCount to 0 step -1
        state$  = eval$("xO" + word$(s$, (tens and (2^i)) + 1))
        yPos$   = str$(17 * j)
        j       = j + 1
        #gbT$   "drawbmp "; led$; " "; state$; " "; yPos$
    next i

    j = 0
    for i = 3 to 0 step -1
        state$  = eval$("xO" + word$(s$, (ones and (2^i)) + 1))
        yPos$   = str$(17 * j)
        j       = j + 1
        #gbO$   "drawbmp "; led$; " "; state$; " "; yPos$
    next i

    #gbT$       "flush currSegment"; sT$
    #gbO$       "flush currSegment"; sO$
    #binClock   "refresh"
end sub
```