

```

;*****;
;*                                     A T C L K                                     *;
;*-----*
;* Task          : This program is a clock driver which can be          *;
;*                used by DOS for functions which access date          *;
;*                and time from the battery powered AT clock.          *;
;*-----*
;* Author        : MICHAEL TISCHER                                       *;
;* Developed on   : 08/04/87                                             *;
;* Last update    : 04/07/95                                             *;
;*-----*
;* Assembly      : MASM ATCLK;                                           *;
;*                LINK ATCLK;                                           *;
;*                EXE2BIN ATCLK ATCLK.SYS                               *;
;*                or                                                    *;
;*                TASM ATCLK                                             *;
;*                LINK ATCLK;                                           *;
;*                EXE2BIN ATCLK ATCLK.SYS                               *;
;*-----*
;* Call          : Copy into root directory, add the command          *;
;*                DEVICE=ATCLK.SYS to the CONFIG.SYS file              *;
;*                and then boot system.                                  *;
;*****;

```

```
code        segment
```

```
        assume cs:code,ds:code,es:code,ss:code
```

```
        org 0                      ;Program has no PSP, therefore
                                   ;it begins at offset address 0

```

```
== Constants =====
```

```
cmd_fld  equ 2                      ;Offset command field in data block
status   equ 3                      ;Offset status field in data block
end_adr   equ 14                    ;Offset driver end addr. in data block
num_db    equ 18                    ;Offset number in data block
b_adr     equ 14                    ;Offset buffer address in data block

```

```
== Data =====
```

```
-- Header of device driver -----
```

```
        dw -1,-1                    ;Connection to next driver
        dw 10000000000001000b       ;Driver attribute
        dw offset strat              ;Pointer to strategy routine
        dw offset intr               ;Pointer to interrupt routine
        db "$CLOCK "                ;New clock driver

```

```
db_ptr    dw (?), (?)                ;Address of data block passed

```

```
mon_tab    db 31                    ;Table with number of days in
february   db 28                    ;the months
           db 31,30,31,30,31,31,30,31,30,31

```

```
== Driver routines and functions =====
```

```
strat      proc far                  ;Strategy routine

        mov  cs:db_ptr,bx            ;Record address of the data block in
        mov  cs:db_ptr+2,es          ;the variable DB_PTR

        ret                          ;Back to caller

```

```
strat      endp

```

```
-----
```

```
intr       proc far                  ;Interrupt routine

        push ax                      ;Store registers on the stack
        push bx
        push cx
        push dx
        push di
        push si

```

```

push bp
push ds
push es
pushf                ;Store the flag register

cld                  ;Increment for string commands

push cs              ;Set data segment register
pop  ds              ;Code is identical with data here

les  di,dword ptr db_ptr;Address of data block to ES:DI
mov  bl,es:[di+cmd_fld] ;Get command code
cmp  bl,4             ;Should time/date be read?
je   ck_read          ;YES --> CK_READ
cmp  bl,8             ;Should time/date be written?
je   ck_write          ;YES --> CK_WRITE
or   bl,bl            ;Should driver be initialized?
jne  unk_fkt           ;NO --> UNK_FKT (Unknown function)

jmp  init             ;Initialize driver

unk_fkt: mov  ax,8003h      ;Code for "unknown command"

;-- Execution completed -----

intr_end label near
or   ax,0100h          ;Set finished bit
mov  es:[di+status],ax ;Store everything in status field

popf                  ;Restore flag register
pop  es               ;Restore other registers
pop  ds
pop  bp
pop  si
pop  di
pop  dx
pop  cx
pop  bx
pop  ax

ret                   ;Back to caller

intr    endp

;-----

ck_read label near      ;Read time/date from the clock

mov  byte ptr es:[di+num_db],6 ;6 bytes are passed
les  di,es:[di+b_adr]   ;ES:DI points to the DOS buffer

mov  ah,4              ;Read function number for date
int  1Ah               ;Call BIOS time interrupt
call date_ofs          ;Change date after offset to 1-1-1980
stosw                   ;Store in buffer

mov  ah,2              ;Read function number for time
int  1Ah               ;Call BIOS time interrupt
mov  bl,ch              ;Store hour in BL
call bcd_bin           ;convert minutes
stosb                   ;Store in buffer
mov  cl,bl              ;Hour to CL
call bcd_bin           ;Convert hour
stosb                   ;Store in buffer
xor  al,al              ;Hundredth of a second is 0
stosb                   ;Store in buffer
mov  cl,dh              ;Seconds to CL
call bcd_bin           ;Convert seconds
stosb                   ;Store in buffer

xor  ax,ax              ;Everything O.K.
jmp  short intr_end     ;Back to caller

;-----

ck_write label near     ;Write time/date into clock

```

```

mov byte ptr es:[di+num_db],6 ;6 bytes are read
les di,es:[di+b_adr] ;ES:DI points to the DOS buffer

mov ax,es:[di] ;Get number of days since 1-1-1980
push ax ;Store number
call ofs_date ;Convert into a date
mov ch,19h ;Year begins with 19..
mov ah,5 ;Set function number for date
int 1AH ;Call BIOS time interrupt

mov al,es:[di+2] ;Get minute from buffer
call bin_bcd ;Convert to BCD
mov cl,al ;Bring to CL
mov al,es:[di+5] ;Get seconds from buffer
call bin_bcd ;Convert to BCD
mov dh,al ;Bring to DH
mov al,es:[di+3] ;Get hours from buffer
call bin_bcd ;Convert to BCD
mov ch,al ;Bring to CH
xor dl,dl ;No summer time
mov ah,3 ;Set function number for time
int 1AH ;Call BIOS time interrupt

;-- Calculate day of the week -----
xor dx,dx ;High word for division
pop ax ;Get number of days from stack
or ax,ax ;Is number 0?
je nodiv ;Yes --> Bypass division
xor dx,dx ;High word for division
mov cx,7 ;Week has seven days
div cx ;Divide AX by 7
nodiv: add dl,3 ;1-1-80 was a Tuesday (Day 3)
      cmp dl,8 ;Is it a Sunday or Monday?
      jb nosomo ;NO --> No correction necessary
      sub dl,cl ;Correct value
nosomo: mov al,6 ;Location 6 in RTC is day of week
      out 70h,al ;Address to RTC address register
      mov al,dl ;Day of the week to AL
      out 71h,al ;Day of the week to RTC data register

xor ax,ax ;Everything O.K.
jmp intr_end ;Back to caller

;-- OFS_DATE: Convert number of days since 1-1-1980 into date -----
;-- Input : AX = Number of days since 1-1-1980
;-- Output : CL = Year, DH = Month and DL = Day
;-- Registers : AX, BX, CX, DX, SI and FLAGS are affected
;-- Info : MON_TAB array converts offsets

ofs_date proc near

      mov cl,80 ;Year 1980
      mov dh,01 ;January
ly: mov bx,365 ;Number of days in a normal year
   test cl,3 ;Is year a leap year?
   jne ly1 ;NO --> LY1
   inc bl ;Leap year has one day more
ly1: cmp ax,bx ;Another year passed?
     jb mo ;NO --> Calculate months
     inc cl ;YES --> Increment year
     sub ax,bx ;Deduct number of days in this year
     jmp short ly ;Calculate next year

mo: mov bl,28 ;Days in February in a normal year
   test cl,11b ;Is the year a leap year?
   jne nulp2 ;NO --> NOLP2
   inc bl ;In leap year February has 29 days
nulp2: mov february,bl ;Store number of days in February

      mov si,offset mon_tab ;Address of months table
      xor bh,bh ;Every month has less than 256 days
mol: mov bl,[si] ;Get number of days in month
     cmp ax,bx ;Another month passed?
     jb day ;NO --> Calculate day

```

```

        sub  ax,bx                ;YES --> Deduct day of the month
        inc  dh                  ;Increment month
        inc  si                  ;SI to next month in the table
        jmp  short mol           ;Calculate next month

day:     inc  al                  ;The remainder + 1 is the day
        call bin_bcd             ;Convert day to BCD
        mov  dl,al              ;Transmit to DL
        mov  al,dh              ;Transmit month to AL
        call bin_bcd             ;Convert to BCD
        mov  dh,al              ;Move to DH
        mov  al,cl              ;Move year to AL
        call bin_bcd             ;Convert to BCD
        mov  cl,al              ;Move to CL

        ret                     ;Back to caller

ofs_date endp

;-- BIN_BCD: Convert binary number to BCD -----
;-- Input      : AL = Binary value
;-- Output     : AL = corresponding BCD value
;-- Registers  : AX, CX and FLAGS are affected

bin_bcd  proc near

        xor  ah,ah              ;Prepare 16 bit division
        mov  ch,10              ;Work in decimal system
        div  ch                  ;Divide AX by 10
        shl  al,1               ;Shift quotient left 4 places
        shl  al,1
        shl  al,1
        shl  al,1
        or   al,ah              ;OR remainder
        ret                     ;Back to caller

bin_bcd  endp

;-- DATE_OFS: Convert Date in number of days since 1-1-1980 -----
;-- Input      : CL = Year, DH = Month and DL = Day
;-- Output     : AX = Number of days since 1-1-1980
;-- Registers  : AX, BX, CX, DX, SI and FLAGS are affected
;-- Info      : MON_TAB array converts date

date_ofs proc near
        call bcd_bin            ;Convert year to binary
        mov  bl,al              ;Transmit to BL
        mov  cl,dh              ;Transmit month to CL
        call bcd_bin            ;Convert Month to binary
        mov  dh,al              ;and transmit again to DH
        mov  cl,dl              ;Transmit day to CL
        call bcd_bin            ;Convert day to binary
        mov  dl,al              ;and transmit again to DL

        xor  ax,ax              ;0 days
        mov  ch,bl              ;Store year
        dec  bl                 ;Decrement one year
year:    cmp  bl,80              ;Counted back to year 1980 ?
        jb   month              ;YES --> convert month
        test bl,11b             ;Is year a leap year ?
        jne  nolpyr             ;NO --> NOLPYR
        inc  ax                 ;A leap year has one more day
nolpyr:  add  ax,365              ;Add days of year
        dec  bl                 ;Back one year
        jmp  short year         ;Process next year

month:   mov  bl,28              ;Days in February in a normal year
        test ch,11b             ;Is current year a leap year?
        jne  nolpyr1            ;NO --> NOLPYR1
        inc  bl                 ;February has 29 days during leap year
nolpyr1: mov  february,bl        ;Store in month table
        xor  ch,ch              ;Every month has less than 256 days
        mov  bx,offset mon_tab  ;Address of month table
month1:  dec  dh                 ;Decrement number of months
        je   add_day            ;All month calculated --> ADD_DAY
        mov  cl,[bx]            ;Get number of days in month

```

```

        add  ax,cx                ;Add to total days
        inc  bx                   ;BX to next month in the table
        jmp  short month1         ;Calculate next month

add_day: add  ax,dx                ;Add current day
        dec  ax                   ;Deduct one day (1-1-80 = 0)
        ret                      ;Back to caller

date_ofs  endp

;-- BCD_BIN: Convert BCD to binary number -----
;-- Input      : CL = BCD value
;-- Output     : AL = corresponding binary value
;-- Registers  : AX, CX and FLAGS are affected

bcd_bin  proc near                ;Convert BCD value in CL to binary
                                         ;Return in AL

        mov  al,cl                ;Transmit value to AL
        shr  al,1                 ;Shift 4 places right
        shr  al,1
        shr  al,1
        shr  al,1
        xor  ah,ah                ;Set AH to 0
        mov  ch,10                ;Process in decimal system
        mul  ch                   ;Multiply AX by 10
        mov  ch,cl                ;Transmit CL to CH
        and  ch,1111b             ;Set high nibble in CH to 0
        add  al,ch                ;Add AL and CH
        ret                      ;Back to caller

bcd_bin  endp

;-----

init      proc near                ;Initialization routine

        ;-- The following code can be overwritten -----
        ;-- by DOS after installing the clock -----

        mov  word ptr es:[di+end_adr],offset init  ;Set end address
        mov  es:[di+end_adr+2],cs                ;of the driver

        mov  ah,9                  ;Display installation message
        mov  dx,offset initm        ;Address of the text
        int  21h                   ;Call DOS interrupt

        xor  ax,ax                 ;Everything O.K.
        jmp  intr_end               ;Back to caller

initm     db 13,10,"**** ATCLK driver installed. (c) 1992 by"
        db " MICHAEL TISCHER",13,10,"$"

init      endp

;-----

code      ends
end

```