

Read and Write Control of the HT1380

D/N : HA0049E

Introduction

This application note applies to the HT48XXX and HT46XXX MCU devices. The HT1380/HT1381 is a standard Holtek device which can implement a calendar and clock function in hardware. The only external component required, to ensure correct operation, is a 32K crystal. It is only necessary for the user to write the initial time and date to the device's corresponding register, after which any data read from the HT1380/HT1381 device will provide the current time and date. The device provides excellent accuracy and convenience.

Driver Description

- Using the Driver
To implement the read and write functions in the HT1380, two drivers are provided. Add the necessary variables in the subroutine to the definition, and add the original file `rw_ht1380.asm` to the program. Modifying the I/O definition can be done directly at the `.section 'data'` by modifying the equ definition.
- Detailed descriptions for each driver
 - Driver Name: `READ_1380`
Function: read data from the HT1380
Entrance argument: none
Exit argument: `acc`
Middle argument: `time_temp, time_count`
Stack: none
 - Driver Name: `WRITE_1380`
Function: read data from the HT1380
Entrance argument: `acc`
Exit argument: none
Middle argument: `time_temp, time_count`
Stack: none

```

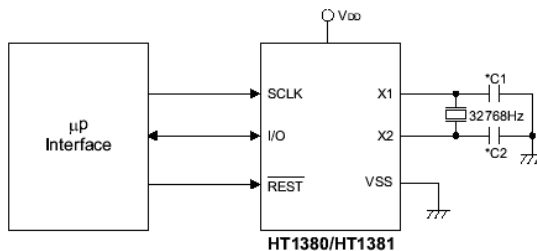
#include ht48r10a-1.inc
rw_ht1380_data .section      'data'
time_temp      db      ?
time_count     db      ?
ht1380_clk     equ     pa.4
ht1380_clk_ctrl equ     pac.4
ht1380_io      equ     pa.5
ht1380_io_ctrl equ     pac.5
ht1380_rest    equ     pa.6
ht1380_rest_ctrl equ     pac.6

rw_ht1380_code1 .section     'code'
;-----
read_ht1380:
    clr     time_temp
    mov     a,8
    mov     time_count,a
    set     ht1380_io_ctrl
read_ht1380_loop:
    clr     c
    set     ht1380_clk
    sz      ht1380_io
    set     c
    rrc     time_temp
    clr     ht1380_clk
    sdz     time_count
    jmp     read_ht1380_loop
    mov     a,time_temp
    ret

;-----
write_ht1380:
    mov     time_temp,a
    mov     a,8
    mov     time_count,a
    clr     ht1380_io_ctrl
    clr     ht1380_io
write_ht1380_loop:
    rrc     time_temp
    sz      c
    set     ht1380_io
    set     ht1380_clk
    nop
    clr     ht1380_clk
    clr     ht1380_io
    sdz     time_count
    jmp     write_ht1380_loop
    ret

```

Circuit Hardware



The following example shows how to use the Driver program. The program's function is to first initialize the HT1380, and then obtain the present value from the HT1380 after one minute and forty seconds. The data is saved in BCD code format.

```

;*****
;FILE NAME:   FRONT PANEL
;MCU:        HT48R10A-1
;MAST OPTION: WDT CLOCK SOURCE: DISABLE WDT
; CLR WDT: ONE
; TIMER CLOCK SOURCE: SYSTEM CLOCK
; WAKE-UP PA: NONE
; INPUT TYPE PA: SCHMITT TRIGGER
; PULL-HIGH: PA,PB,PC
; BZ/BZB: ALL DISABLE
; LVR: DISABLE
; OSC: EXT. CRYSTAL
; FOSC: EXTERNAL
; SYSVOLT: 5.0V
; SYSFREQ: 4MHZ
; PWM: DISABLE
; PFD: DISABLE
;AUTHOR: RADOME
;HISTORY: 2003.09.17
;*****
include Ht48r10a-1.inc
    PUSH macro
    mov  acc_bk,a
    mov  a,status
    mov  status_bk,a
    endm

    POP macro
    mov  a,status_bk
    mov  status,a
    mov  a,acc_bk

```

```

        endm
;-----
ht1380_clk      equ    pa.4
ht1380_clk_ctrl equ    pac.4
ht1380_io       equ    pa.5
ht1380_io_ctrl  equ    pac.5
ht1380_rest     equ    pa.6
ht1380_rest_ctrl equ    pac.6
;*****
FrontPanel_data.section 'data'
;*****
;System
acc_bk          db     ?
status_bk       db     ?

;ht1380
second          db     ?
minute          db     ?
hour            db     ?
date            db     ?
month           db     ?
day             db     ?
yearh           db     ?
yearl           db     ?
time_count      db     ?
time_temp       db     ?

;BCD/HEX
data_bcd        db     ?
data_hex        db     ?
data_count      db     ?
data_temp       db     ?

f_test          dbit
;*****
FrontPanel_code.section 'code'
;*****
        org    0000h
        jmp    main

        org    0004h          ;External Interrupt
        reti

        org    0008h          ;Timer Interrupt
timer_int:
        push          ;macro program for interrupt protection

        inc    data_temp
        mov    a,data_temp

```

```

sub    a,250
snz    c
jmp    timer_end
clr    data_temp

inc    data_count
mov    a,data_count
sub    a,50
snz    c
jmp    timer_end
set    f_test

timer_end:
pop                                ;macro program for interrupt and return
                                        ;the relative register to its original
                                        ;status
retl
;*****
;Initializers
;*****
main:
clr    wdt
clr    intc
clr    tmrc

clr    pa
clr    pac
clr    pb
clr    pbc
clr    pc
clr    pcc                                ;initialize the program

mov    a,20h
mov    mp,a
mov    a,64
clr    iar
inc    mp
sdz    acc
jmp    $-3                                ;clear the RAM

mov    a,00000101b
mov    intc,a
mov    a,6                                ;8ms
mov    tmr,a
mov    a,10000110b
mov    tmrc,a
set    tmrc.4
nop
nop

```

```

nop
clr    tmrc.4
mov    a,6                ;8ms
mov    tmr,a
mov    a,10010110b
mov    tmrc,a

mov    a,00h
mov    second,a
mov    a,59h
mov    minute,a
mov    a,23h
mov    hour,a
mov    a,30h
mov    date,a
mov    a,09h
mov    month,a
mov    a,02h
mov    day,a
mov    a,03h
mov    yearl,a
call   init_ht1380        ;write in 03-09-30 23:59:00

snz    f_test
jmp    $-1
call   get_time
jmp    $                  ;read the value after 1 min and 40
                           ;seconds

;*****
;ht1380
;*****
;-----
;Initialize ht1380
;-----
init_ht1380:
clr    ht1380_rest_ctrl
clr    ht1380_clk_ctrl
clr    ht1380_io_ctrl

clr    ht1380_rest
nop
set    ht1380_rest
mov    a,10001110b
call   write_ht1380
mov    a,00000000b
call   write_ht1380      ;disable te write protect
clr    ht1380_rest
nop

```

```

set    ht1380_rest
mov    a,10111110b      ;burst mode command
call   write_ht1380
mov    a,second         ;"CH" bit set 0
call   write_ht1380
mov    a,minute
call   write_ht1380
mov    a,hour
call   write_ht1380
mov    a,date
call   write_ht1380
mov    a,month
call   write_ht1380
mov    a,day
call   write_ht1380
mov    a,year1
call   write_ht1380
clr    ht1380_rest
ret

;-----
;Write ht1380
;-----
write_ht1380:
    mov    time_temp,a
    mov    a,8
    mov    time_count,a
    clr    ht1380_io_ctrl
    clr    ht1380_io
write_ht1380_loop:
    rrc    time_temp
    sz     c
    set    ht1380_io
    set    ht1380_clk
    nop
    clr    ht1380_clk
    clr    ht1380_io
    sdz    time_count
    jmp    write_ht1380_loop
    ret

;-----
;Get time
;-----
get_time:
    clr    ht1380_rest_ctrl
    clr    ht1380_clk_ctrl
    clr    ht1380_io_ctrl

    clr    ht1380_rest
    nop

```

```

set    ht1380_rest
mov    a,10111111b      ;burst mode command
call   write_ht1380
nop
call   read_ht1380
mov    second,a
call   read_ht1380
mov    minute,a
call   read_ht1380
mov    hour,a
call   read_ht1380
mov    date,a
call   read_ht1380
mov    month,a
call   read_ht1380
mov    day,a
call   read_ht1380
mov    year1,a
clr    ht1380_rest
ret

;-----
;Read ht1380
;-----
read_ht1380:
    clr    time_temp
    mov    a,8
    mov    time_count,a
    set    ht1380_io_ctrl
read_ht1380_loop:
    clr    c
    set    ht1380_clk
    sz    ht1380_io
    set    c
    rrc    time_temp
    clr    ht1380_clk
    sdz    time_count
    jmp    read_ht1380_loop
    mov    a,time_temp
    ret

;*****
;BCD&HEX
;*****
;-----
;BCD to HEX
;-----
bcd2hex:
    swapa data_bcd
    and    a,0fh
    rl    acc

```

```
        mov    data_temp,a
        rl     acc
        rl     acc
        addm  a,data_temp
        mov   a,data_bcd
        and   a,0fh
        add   a,data_temp
        mov   data_hex,a
        ret

;-----
;HEX to BCD
;-----
hex2bcd:
        clr   data_bcd
        mov   a,8
        mov   data_count,a
hex2bcd_loop:
        rlc   data_hex
        mov   a,data_bcd
        adc   a,data_bcd
        daa   data_bcd
        sdz   data_count
        jmp   hex2bcd_loop
        ret
;*****
end
```