HT1380 的读写控制

文件编码：HA0049s

HT1380/HT1381 是一款用硬件来实现日历及时钟功能的标准品,在与其它 MCU 配套使用时外部只需挂一颗 32K 的晶振。用户使用时只需将初始时间日期写入其相应的寄存器内即可，随后从 HT1380/HT1381 内读出的数据即为当前时间日期值。它精度高，而且应用非常简单方便。

DRIVER 的使用说明：

一，driver 的使用

针对 HT1380 的读写，共提供了二个 driver，实现对 HT1380 进行读出和写入的功能。使用时将该子程序中要用到的变量加入到你的定义中，将汇编源文件 rw_ht1380.asm 加入到你的程序中即可。要修改 I/O 的定义可以直接在 .section ‘ data ’ 中修改 equ 的定义。

二，各个 driver 的详细说明

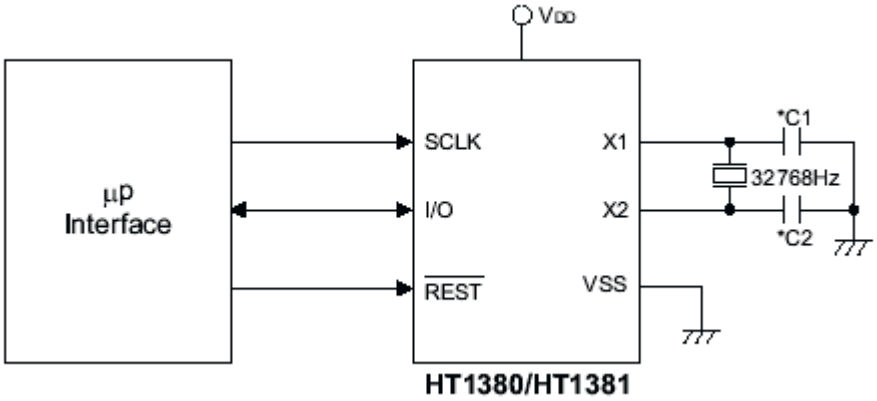
1，driver 名称： READ_1380
实现功能： 从 HT1380 中读出一个数据
入口参数： none
出口参数： acc
中间变量： time_temp, time_count
堆栈使用： 无

2，driver 名称： WRITE_1380
实现功能： 向 HT1380 中写入一个数据
入口参数： acc
出口参数： none
中间变量： time_temp, time_count
堆栈使用： 无

```
#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
```

硬件电路图如下：



下面用实例来说明该怎样使用此 driver 程序。
该程序动作是：先对 HT1380 赋初值，过 1 分 40 秒后再去从 HT1380 里去取值。取出数据以 BCD 码格式存放.

```
;*****************************************
;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
    PUSHmacro
    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_ctrlequ 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                        ;宏程序，做进入中断保护

    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                             ;宏程序，做出中断恢复相关寄存器的原状态
    reti
;*****************************************
;Initializers
;*****************************************
main:
    clr     wdt
    clr     intc
    clr     tmrc

    clr     pa
    clr     pac
    clr     pb
    clr     pbc
    clr     pc
    clr     pcc                 ;程序初始化

    mov     a,20h
    mov     mp,a
    mov     a,64
    clr     iar
    inc     mp
```

```
    sdz     acc
    jmp     $-3                 ;清 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
    callinit_ht1380         ;写入 03-09-30  23:59:00

    snz     f_test
    jmp     $-1
    call    get_time
    jmp     $                   ;1 分 40 秒后读出来的值

;*****************************************
;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
    callwrite_ht1380
    mov     a,00000000b
    callwrite_ht1380    ;disable te write protect
    clr     ht1380_rest
    nop
    set     ht1380_rest
    mov     a,10111110b     ;burst mode command
    callwrite_ht1380
    mov     a,second    ;"CH" bit set 0
    callwrite_ht1380
    mov     a,minute
    callwrite_ht1380
    mov     a,hour
    callwrite_ht1380
    mov     a,date
    callwrite_ht1380
    mov     a,month
    callwrite_ht1380
    mov     a,day
    callwrite_ht1380
    mov     a,yearl
    callwrite_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
    callwrite_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     yearl,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
    addma,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
```