HT48 MCU 对 HT1621 LCD 控制器的使用

文件编码：HA0018s

介绍：

HT1621 是一款 128 个位元的 LCD 控制器件，内部 RAM 直接对应 LCD 的显示单元。相应的软件使它适用于包括 LCD 模块和显示子系统在内的多功能应用。主控制器与 HT1621 接口只需 4 到 5 根线。内置的省电模式极大的降低了功耗。本文介绍用 HT48R30A-1 单片机来控制 HT1621,并介绍如何点亮及清除 LCD 所有位元。

原理：

对于 HT1621，操作之前应该给它发送标志码，表明要求工作在哪种状态。标志码的定义如下表：

操作	状态	标志码
读	数据	110
写	数据	101
读-修改-写	数据	101
控制	命令	100

为了点亮 LCD，必须先给出两个控制指令：SYSTEM ENABLE 和 LCD ON。SYSTEM ENABLE 指令码是：10000000001X（X 为 Don't care bit)。LCD ON 指令码是：10000000011X（X 为 Don't care bit)。操作结束可以用 SYSTEM DISABLE 来关闭 LCD。

由于是串行通信，数据应该先出现在 DATA INPUT 脚，然后给出一个写允许信号（WR）,输入一位数据，接着输入第二位…直到全部写入。

对 RAM 区不连续写数据过程是这样的：

1	0	1	A5	A4	A3	A2	A1	A0	D0	D1	D2	D3	结束位	下一个过程

先发送标志码 101，表明下面要进行写操作。然后发送地址码 A5~A0，用 D0~D3 指定对应的位，就可以对 LCD 相应的位元操作了。读的过程除了标志位不同，其余类似。

连续读写时，给出起始地址，操作结束地址自动加一。

例程：

本例介绍如何点亮和清除 LCD 全部位元。程序流程如下：

系统初始化→1621 启动→清除 LCD 全部位元→点亮 LCD 全部位元→读出某一单元的值进行比较→系统初始化

电路图：参照 HT1621 的规格说明书

```
;1621driver.asm
;这个程序是用 HT48R30A-1 去控制 HT1621
;控制口的结构：
;       PB1 -- datum
;       PB2 -- WRB
;       PB3 -- CSB
;       PB4 -- RDB
```

```
;       OSC: Ext. Crystal
;       WDT clock source: Disable WDT
;       input type PA:      Sshmitt Trigger
;       Pull-high PA:       Pull-high PA
;       Pull-high PB:       Pull-high PB
;       BZ/BZB  :           BZ ENABEL/BZB DISABLE
;        Fsys:              4M
;注意：在写程序时，时序一定要给正确

include    ht48r30a-1.inc

csb        equ  pb.3
csbc       equ  pbc.3
wrb        equ  pb.2
wrbc       equ  pbc.2
datum      equ  pb.1
datumc     equ  pbc.1
rdb        equ  pb.4
rdbc       equ  pbc.4
lig        equ  pc.3

;-----------------------------
num_mem      equ  [7fh]
;-----------------------------
;宏定义
;延迟宏，延迟 5 微秒
d_1        macro
      jmp  $+1
      jmp  $+1
      nop
endm
;-----------
lcddriver .section    'data'
count           db    ?             ;用作记录循环次数
code_datum      db    ?             ;command code or memory datum bits
code_datum1     db    ?             ;only used in read_modify_write mode
mem_addr        db    ?             ;memorty address for selecting segment

temp_da         db    ?
t_addr_h        db    ?             ;just a buffer
;-----------------------------

lock   .SECTION 'CODE'
```

```
        org     00h

        jmp     start
        org     04h
        reti
        org     08h
        reti

start:
        clr     pb                  ;Initial
        set     csb
        clr     pbc
        set     pbc.0
        clr     pc
        clr     pcc
        clr     intc

        mov     a, 50h
        mov     num_mem, a
        mov     a, 20h
        mov     mp0, a

clr_ram:                            ;initial ram
        clr     r0
        inc     mp0
        sdz     num_mem
        jmp     clr_ram
;----------------------------------------
ini_status:
        mov     a, 87h
        mov     tmrc, a

show_k:
        set     lig
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        mov     a, 0e3h                 ;NORMAL
        mov     code_datum, a
        call    send_command
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        mov     a, 01h                  ;SYS ENABLE
        mov     code_datum, a
        call    send_command
LO:
```

```
        mov     a, 029h             ;4com;1/3bias
        mov     code_datum, a
        call    send_command

        mov     a, 3                ;LCD On
        mov     code_datum, a
        call    send_command
        call    clr_lcm             ;cls lcd
        jmp     $+1
        jmp     $+1
        call    show_lcm            ;light all dots

        clr     code_datum          ;reading then writing in the same address
        mov     a, 4
        mov     mem_addr, a
        call    read
        mov     a, 07h
        xor     a, code_datum
        snz     z
        jmp     error

        mov     a, 2
        mov     code_datum, a
        call    send_command
        jmp     $+1
        jmp     start               ;do it repeat
error:
        jmp     $
;*****************************************
;Purpose   :   send command
;Parameter:
;           code_datum : byte
;Return    :          none
;Modified :    acc, status
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
send_command    proc
        clr     CSB
        clr     datumC

        set     datum
        clr     WRB                 ;COMMAND ID '100'
        d_1
        set     WRB                  ;1
```

```
        nop
        clr     datum                   ;00
        clr     WRB
        d_1
        set     WRB
        nop
        CLR     WRB
        d_1
        set     WRB

        mov     A, 8                        ; send code
        mov     count, A
LOOP1:
        clr     datum
        sz      code_datum.7
        set     datum
        rl      code_datum
        clr     WRB
        d_1
        set     WRB
        sdz     count
        jmp     loop1

        clr     WRB
        d_1
        set     WRB
        nop
        set     CSB             ;close csb signal,not selecting the chip
        ret
send_command    endp
;-------------------------------
;Purpose    :    write datum to 1621
;Parameter:
;       code_datum    : byte
;      mem_addr       : byte
;Return     :          none
;Modified :     acc, status
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
write:
        clr     CSB
        clr     datumc

        set     datum
```

```
        clr     WRB             ;WRITE mode ID '101'
        d_1
        set     WRB

        clr     datum
        clr     WRB
        d_1
        set     WRB

        set     datum
        clr     WRB
        d_1
        set     WRB

        mov     a, 6
        mov     count, a
writeloop1:
        clr     datum
        sz      mem_addr.5      ;sending memory address for selecting segment
        set     datum
        clr     WRB
        d_1
        set     WRB
        rl      mem_addr
        sdz     count
        jmp     writeloop1

        mov     a, 4
        mov     count, a
writeloop2:
        clr     datum
        sz      code_datum.0    ;sending memory content for deciding comments's state
        set     datum
        clr     WRB
        d_1
        set     WRB
        rr      code_datum
        sdz     count
        jmp     writeloop2
        set     CSB
        ret
;-------------------------------------
;Purpose   :   read datum from 1621
```

```
;Parameter:
;       mem_addr      : byte
;Return      :
;       code_datum    : byte
;Modified :      acc, status
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
read proc
            clr         CSB
            clr         datumc

            set         datum
            clr         WRB                 ;READ mode ID '110'
            d_1
            set         WRB

            clr         WRB
            d_1
            set         WRB

            clr         datum
            clr         WRB
            d_1
            set         WRB

            mov         a, 6
            mov         count, a
readloop1:
            clr         datum
            sz          mem_addr.5          ;sending memory address for selecting segment
            set         datum
            clr         WRB
            d_1
            set         WRB
            rl          mem_addr
            sdz         count
            jmp         readloop1

            set         datumc
            mov         a, 4
            mov         count, a
readloop2:
            clr         RDB
            d_1
```

```
            set     RDB
            rr      code_datum
            clr     code_datum.3
            sz      datum               ;sending memory content for deciding comments's state
            set     code_datum.3
            sdz     count
            jmp     readloop2

            mov     a, 0fh
            andm    a, code_datum

            set     CSB
            ret
read endp
;----------------------------------------
;Purpose   :   read datum from 1621, then write a datum in the same register
;Parameter:
;mem_addr    : byte
;Return    :   none
;Modified :    acc, status
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 rm_write     proc
            clr     CSB
            clr     datumc

            set     datum               ;READ-MODIFY-WRITE mode ID   '101'
            clr     WRB
            d_1
            set     WRB

            clr     datum
            clr     WRB
            d_1
            set     WRB

            set     datum
            clr     WRB
            d_1
            set     WRB

            mov     a, 6
            mov     count, a
rmwloop1:
```

```
        clr     datum
        sz      mem_addr.5          ;sending memory address for selecting segment
        set     datum
        clr     WRB
        d_1
        set     WRB
        rl      mem_addr
        sdz     count
        jmp     rmwloop1

        set     datumc
        mov     a, 4
        mov     count, a
rmwloop2:
        clr     RDB
        d_1
        set     RDB
        rr      code_datum1
        clr     code_datum1.3
        sz      datum               ;read memory content out
        set     code_datum1.3
        sdz     count
        jmp     rmwloop2

        mov     a, 0fh
        andm    a, code_datum1

        clr     datumc
        mov     a, 4
        mov     count, a
        mov     a, temp_da
        andm        a, code_datum
rmwloop3:
        clr     datum
        sz      code_datum.0        ;sending memory content for deciding comments's state
        set     datum
        clr     WRB
        d_1
        set     WRB
        rr      code_datum
        sdz     count
        jmp     rmwloop3
        set     CSB
```

```
        ret
rm_write endp
;------------------------------
show_lcm:
        mov     a, 00h
        mov     t_addr_h, a
        mov     a, 31h
        mov     num_mem, a
clr_n1:
        mov     a, t_addr_h
        mov     mem_addr, a
        mov     a, 07h
        mov     code_datum, a
        call    write
        inc     t_addr_h
        sdz     num_mem
        jmp     clr_n1
        ret
;------------------------------
clr_lcm:
        mov     a, 00h
        mov     t_addr_h, a
        mov     a, 31h
        mov     num_mem, a
clr_n:
        mov     a, t_addr_h
        mov     mem_addr, a
        mov     a, 0h
        mov     code_datum, a
        call    write
        inc     t_addr_h
        sdz     num_mem
        jmp     clr_n
        ret
```