直接用单片机驱动 LCD，以求省电省钱

北京中显电子有限公司诚意奉献

LED 数码管的驱动是比较简单也容易理解的，多位数码管一般是 LED 阵列的形式，每个数字使用一个公共端，不同数字的对应同笔段使用一个控制端；驱动采用分时扫描没个数字位，动态显示。但是 LED 比较费电，想做一个用电池供电的钟，用发光管电池就撑不了多久了。于是考虑用液晶。

现有一个 4 位笔段式液晶屏，4 个数字最中间有冒号，边上还有几个箭头符号，一共有 15 个引脚，正合适用 AVR 来驱动做一个钟。

笔段式 LCD 屏的结构与 LED 数码管很相似，但是由于是液晶，工作机理上不同，驱动方式也有很大差异：

(1) LED 有正负之分，液晶笔划没有。

(2) LED 在直流电压下工作，液晶需要交流电压，防止电解效应。

(3) LED 需要电流提供发光的能量，液晶笔划显示状态下电流非常微弱。

(4) LED 对微小电流不反应，液晶则很敏感。

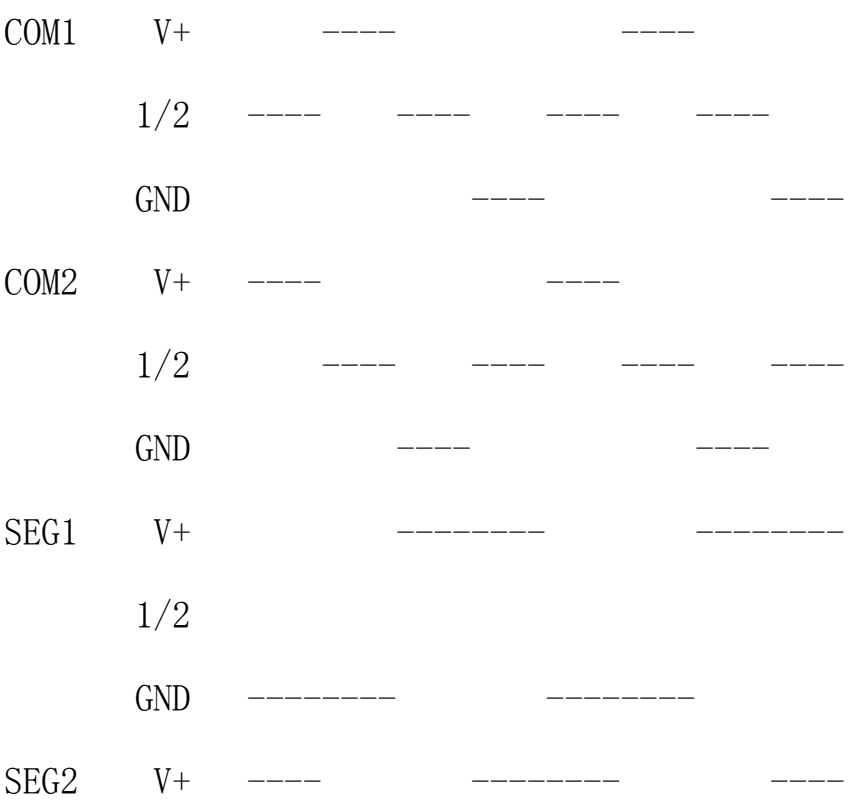
不难看出，用 LED 的驱动方式来对待 LCD 屏是行不通的。一般人在买回来测试这块液晶屏之前没有意识到，于是走了不少的弯路。与 LED 驱动不同的是需要给每个笔划加上一个交流电压。一般用 30-60Hz 的方波就可以了，频率再低显示会有所波动，频率高了功耗也会增加，因为 LCD 对电路呈现容性。而且，正负电压都可以“点亮”液晶。

好在 AVR 的 I/O 口可以三态输出，也就是除了高/低电平，还可

以呈现高阻抗，相当于断开连接。于是想到了这样的办法：不需要显示的那一组笔划对应的公共端悬空(I/O 口选择三态)，那么就不会加上电压了。照这个思路，把实验电路焊好，出来的显示却是一团糟：笔划都黑了看不清。于是考虑到液晶本身的问题：阻抗高，而且有电容，是不可一边悬空的！这个道理也许跟 CMOS 输入端差不多。查找了一些关于液晶的资料，大致知道 LCD 屏不是那么简单的，驱动方式通常是 1/N（占空比），也就是电压不止高低两档。可是单片机 I/O 没有那么多输出状态可以选择。

1/2 Bias 驱动

不显示的液晶笔划两端电压相等，显示的不等。这样一个要求在扫描方式下不能满足，于是改为电压等级不同。1/2 Bias 驱动就是这样的，如下：



1/2

GND ———————— ————————

如此，在 COM1,SEG1 选择的笔划上，加上的电压为 -1/2, -1, +1/2, +1 ... 在 COM1,SEG2 选择的笔划上，加上的电压为 +1/2, -1, -1/2, +1 ... 在 COM2,SEG1 选择的笔划上，加上的电压为 -1, -1/2, +1, +1/2 ...在 COM2,SEG2 选择的笔划上，加上的电压为 0, -1/2, 0, +1/2 ...

计算一下大致的平均功率(如果液晶灰度与电压平方成正比，实际不是这样)前三者是一样的，都是 1+(1/2)^2=5/4, 对于最后一个 0+(1/2)^2=1/4 因此显示的功率比为 5:1, 显示状态会是这样：

SEG1 SEG2

: :

COM1 - - - 0 - - - 0

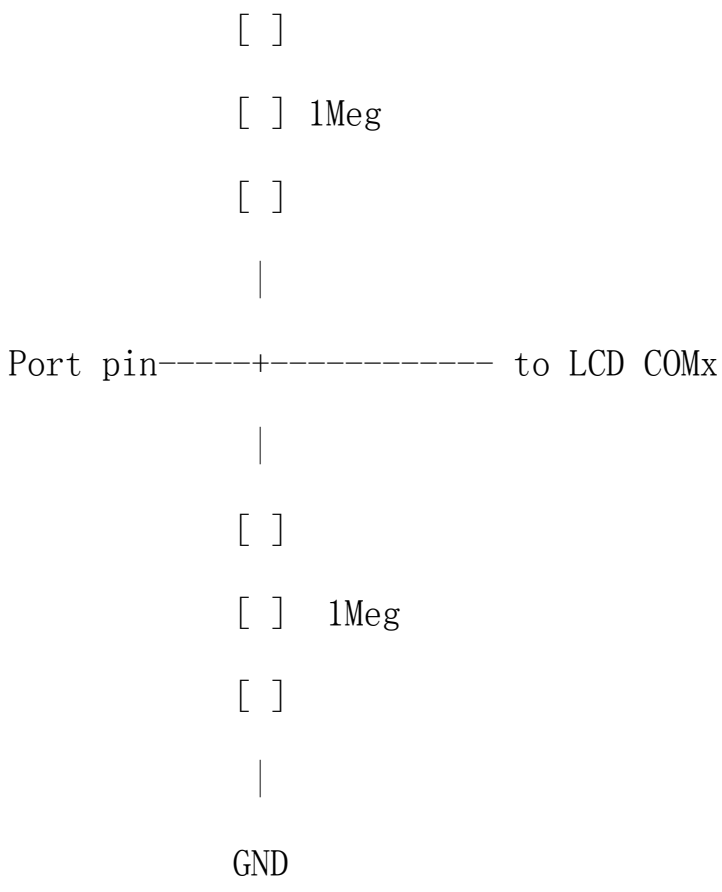
: :

COM2 - - - 0 - - - x

AVR I/O 没有能力输出 1/2 Vcc 的电压(ADC 在这里就不要考虑了，浪费 I/O 还不如用静态液晶屏)，因此没有办法实现真正的 1/2 Bias 驱动。但是注意到要提供一个一半电源电压也不是难事，既然 AVR I/O 口可以三态，可以用两个电阻分压将端口“拉”到 1/2 Vcc 就好了，于是，1/2 Bias 驱动的做法可以这样：

Vcc

|



取电阻 1Meg 是综合耗电与分压效果考虑的。这样在 COMx 就可以产生三种电压值，就达到了 1/2 Bias 动态驱动的目的。实现起来在前面的基础上增加电阻即可，我的屏有 4 个公共端，因此用了 8 个电阻，数字就能够显示出来了。

虽然显示的确做到了,然而效果却不能让人满意。具体表现就是需要正对着 LCD 屏看才是很清晰的;如果斜着看,就可能一片混浊了，没有达到实用。用 2 节 Ni-MH 供电时候正着看没问题，用 2 节干电池(电压提高一点)就不是很清晰了。如前面的分析，那些没有被选择的笔段其实也加上了变化的电压，只不过与选择的比段相比电压平均有效值低一些。这两个的差异足够显著，才能保证显示效果。

再分析 1/2 Bais 驱动在 LCD 屏上 1/4 分时扫描的结果：一个周期内，“点亮”的笔段平均功率=1^2+(1/2)^2+(1/2)^2+(1/2)^2=7/4，而没有被“点亮”的笔段为=0+(1/2)^2+(1/2)^2+(1/2)^2=3/4，两者之比 7:3

跟前面的例子分析对比看出，从 1/2 分时扫描变到 1/4 分时扫描，显出来的笔段和不显的笔段上，电压产生平均功率的对比从 5:1 变到 7:3 了。于是尝试从软件上改变扫描时序，也不能改进显示效果，看来 1/2 Bias 不够用的了。

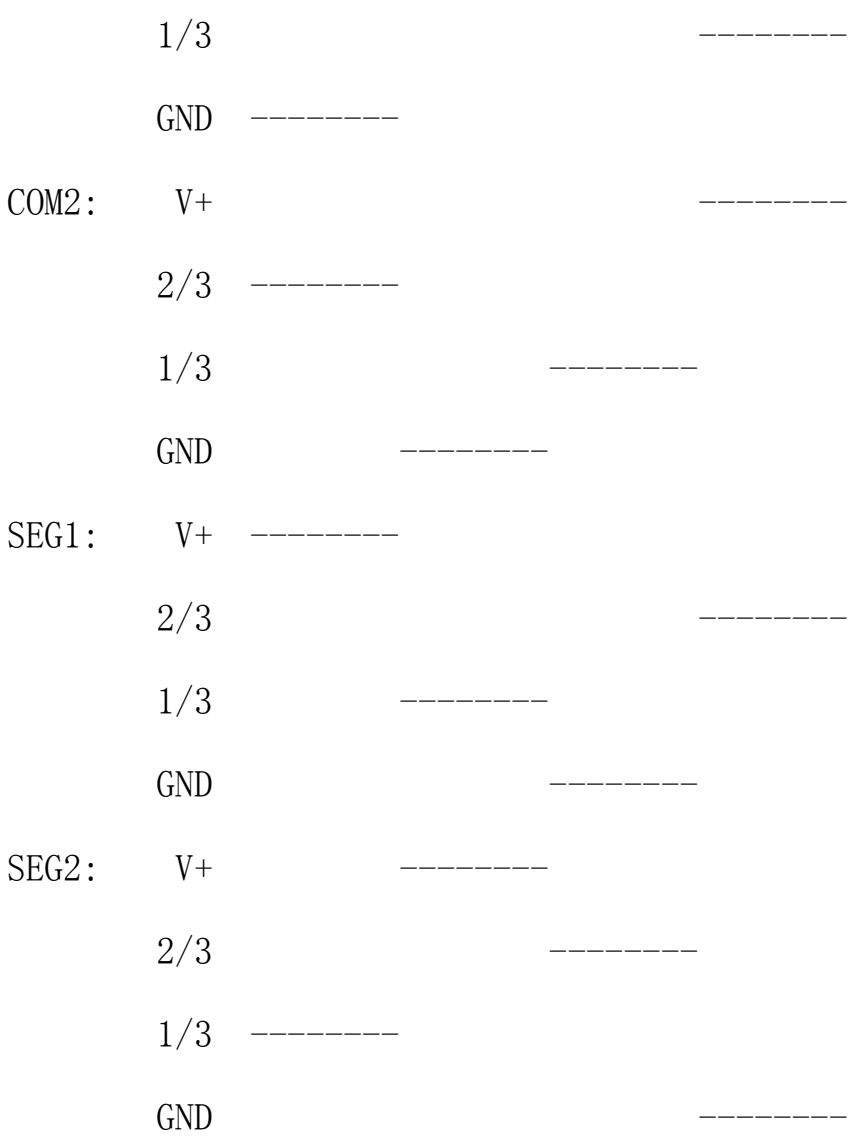
于是查询了 Nokia 3310 液晶手册其中对于 LCD 电压输出时序的描述。恰好里面有一个图，绘出了行和列控制线上的波形。从坐标轴上看出 Vlcd 和 Vss 之间另外还有 4 个电压等级。这么多种电压用 AVR I/O 实现已经不现实了。

于是再考虑选用带有 LCD 驱动功能的 MCU，AVR 只有一款 ATmega169，封装形式不适合 DIY。Microchip 有一款 PIC16F913，有 28DIP 的封装，看上去正合适。暂时不知道价格，于是先找来它的手册看看。详细看了 LCD 驱动模块的部分，发现 PIC16F913 也只有 1/2 Bias 驱动和 1/3 Bias 驱动两种选项，分时最多为 1/4 分时驱动，对于手里的屏正好。

1/3 Bias 驱动需要将 Vcc--GND 之间的电压三等分，一个周期驱动波形示例如下：

COM1: V+ --------

2/3 --------



在 (COM1,SEG1) 笔段上，电压为 +1, -1/3, -1, +1/3 ... 在 (COM1,SEG2) 上为 +1/3, +1/3, -1/3, -1/3 ... 在 (COM2,SEG1) 上 : +1/3, +1/3, -1/3, -1/3 ... 在 (COM2,SEG2) 上 : -1/3, +1, +1/3, -1 ...

于是计算平均功率，在 (COM1,SEG1) 和 (COM2,SEG2) 上面是 2*1^2+2*(1/3)^2=20/9 在 (COM1,SEG2) 和 (COM2,SEG1) 上面是 4*(1/3)^2=4/9, 两者之比 5:1

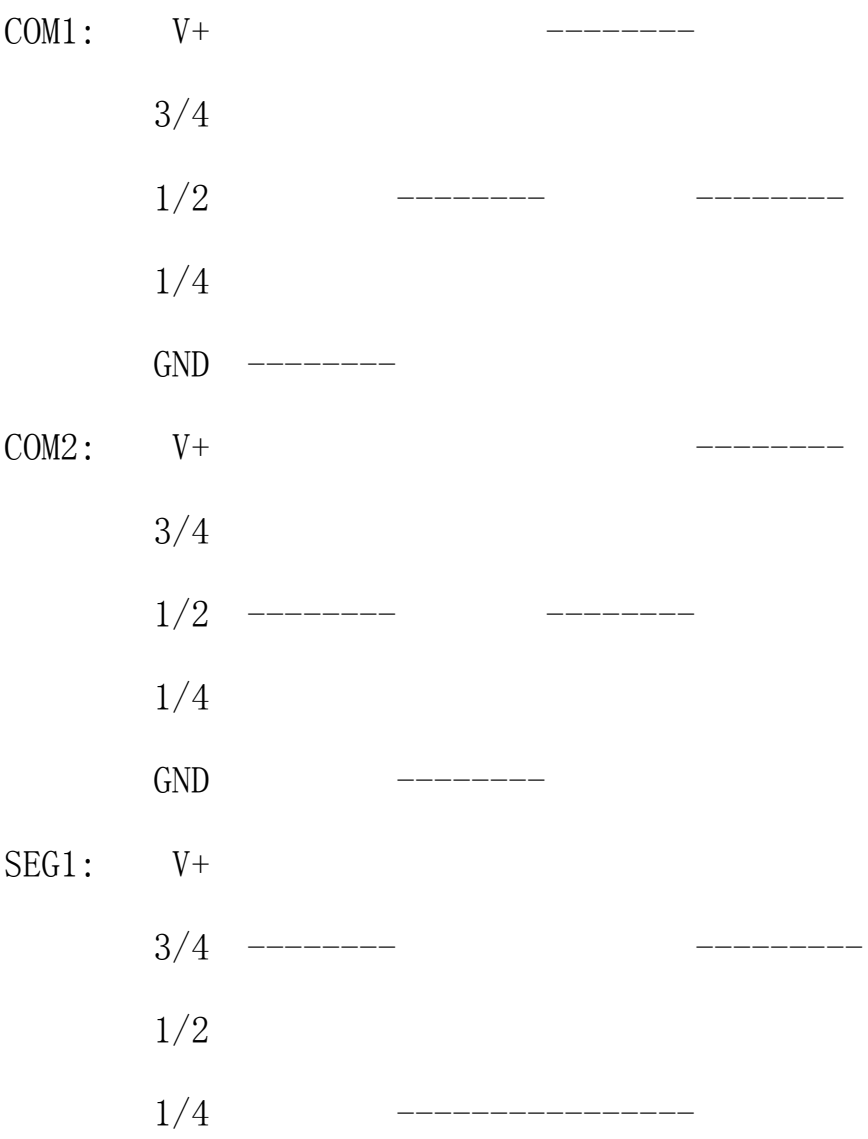
假如不是上图的 1/2 分时驱动而是 1/4 分时驱动，这个比例将

变为

2*1^2+6*(1/3)^2 vs 8*(1/3)^2 = 3:1

若将原来的 1/2 Bias 改用 1/3 Bias 驱动，对于我的 LCD 屏这个比值从 7:3 改善为 3:1 了。既然 PIC16F913 只设计了 1/2 Bias 与 1/3 Bias，用起来应该问题不大。

AVR 单个 I/O 口要实现 4 种电压输出——不可能吧，总之是想不出来了。AVR 最多只有三种电压输出，能不能对这个电压再做等分呢？一番思索之后决定就 4 等分吧.



GND

SEG2: V+

3/4 ----------

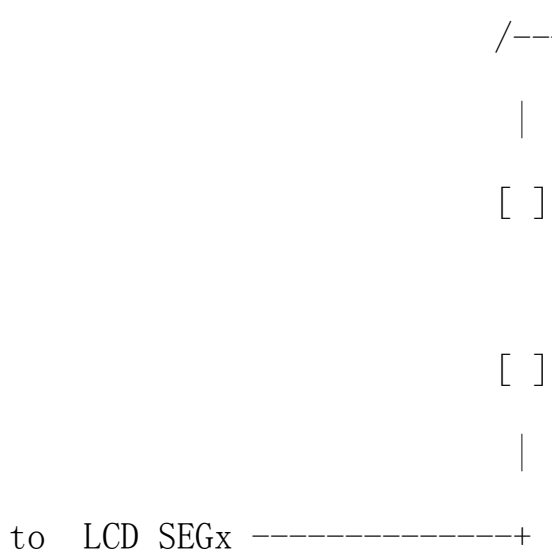
1/2

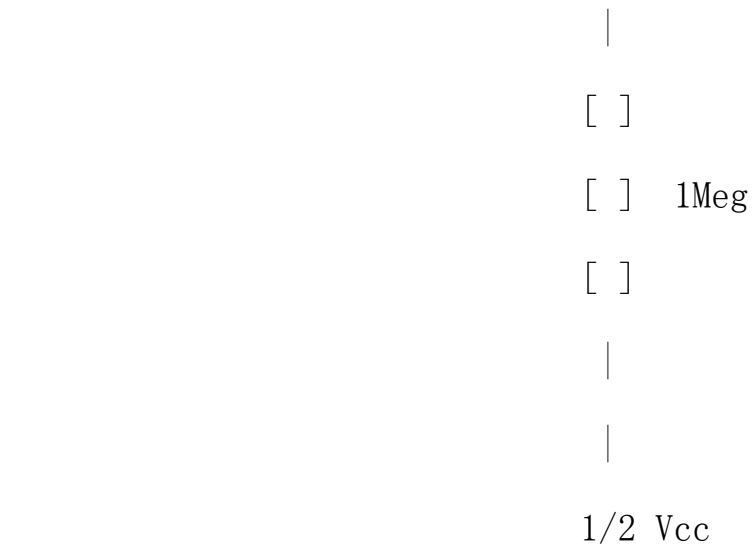
1/4 -------- ---------

GND

实际做法就是 SEGx 输出有两种：3/4*Vcc 和 1/4*Vcc, 而 COMy 输出有三种：Vcc, GND, 1/2*Vcc. 对于每个 I/O 口，并不需要 4 种电压输出。当然这样跟 1/3 Bias 驱动是不一样的，但是却达到了 1/3 Bias 驱动的效果，只不过加在液晶笔段上的电压绝对值最大不是 Vcc 而是 3/4*Vcc 了，因此电源电压也需要提高。这里计算省略。

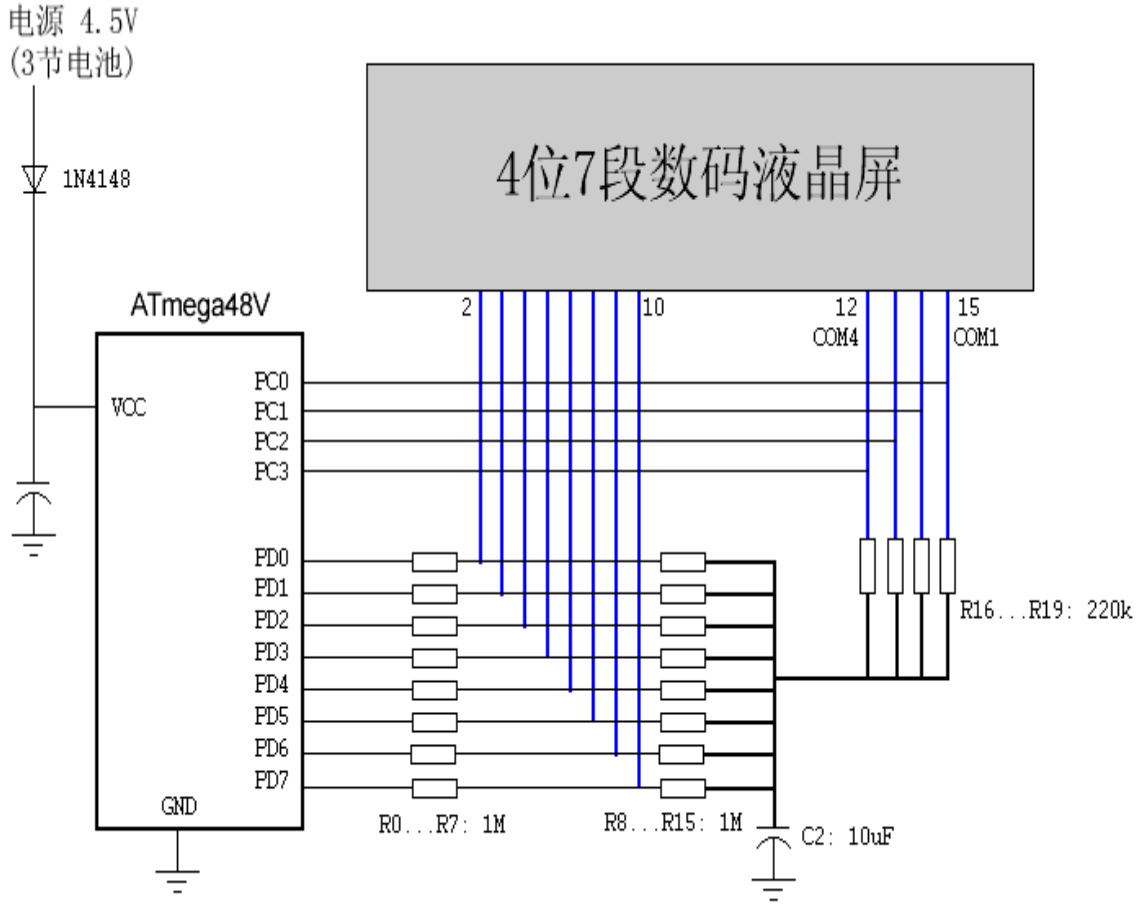
这种驱动方式称之为"伪 1/3 Bias 驱动". 对于 COMy 的处理和前面一样，对于 SEGx, 将 I/O 输出电压改变一下，高电平 3/4*Vcc, 低电平 1/4*Vcc 就好了。具体做法是：

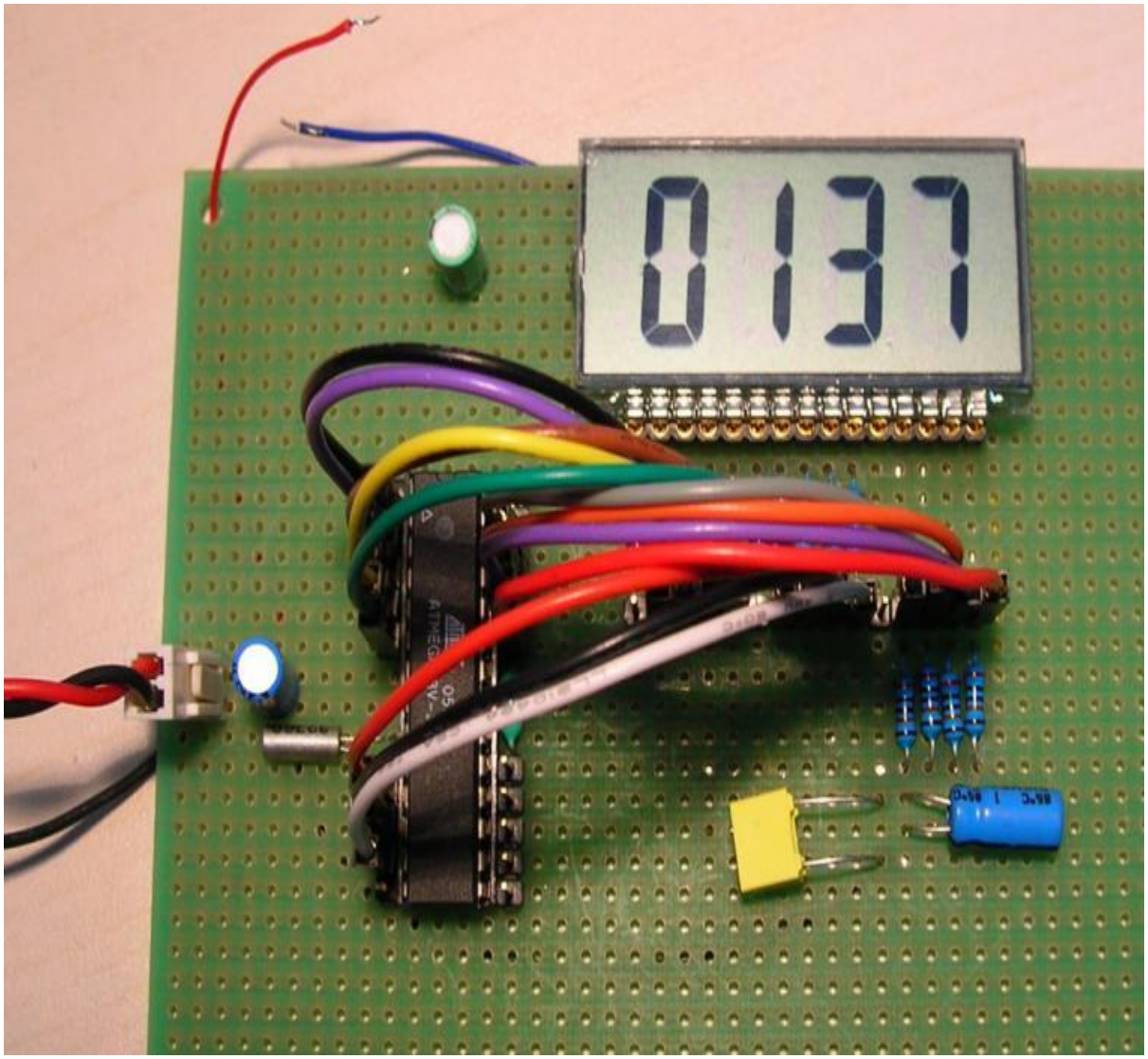




这里的 1/2 Vcc 可以将电源电压用电阻分压得到，想的办法是直接接个几 uF 电容到 GND，实验是成功的。因为随着扫描的进行，这个地方的平均电压是输出高电平和低电平的一半。

下面做了一个 Mega48V 的秒计数器，再改改就能把钟做出来了。


电源 4.5V
(3节电池)
1N4148
4位7段数码液晶屏
ATmega48V
VCC
GND
PC0
PC1
PC2
PC3
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
2
10
12
COM4
15
COM1
R0...R7: 1M
R8...R15: 1M
R16...R19: 220k
C2: 10uF



程序：汇编语言

Timer2 用外接 32768 晶振提供时钟，整个系统耗电大约 30 微安。

```
; lcddisplay.asm

; Test raw LCD display

.include "m48def.inc"

.org 0x0000

    rjmp start

.org OC2Aaddr

    rjmp isr_timer2
```

```
.org 0x0020
table:
    .DB 0b11101101, 0b00101000, 0b10110101, 0b10111001
    .DB 0b01111000, 0b11011001, 0b11011101, 0b10101000
.DB 0b11111101, 0b11111001
start:
    ldi r16, 1<<PUD
out MCUCR, r16 ; disable all I/O pull-up
    ldi r16, 1<<AS2
sts ASSR, r16 ; enable asynchronous mode
    ldi r16, 1<<WGM21
sts TCCR2A, r16          ; CTC mode
ldi r16, 31
sts OCR2A, r16 ; preset compare A
    ldi r16,1<<CS21 ; divide by 8
;    ldi r16,1<<CS20 ; use clkIO as source
sts TCCR2B, r16
ldi r16, 1<<OCF2A
out TIFR2, r16 ; clear flag
ldi r16, 1<<OCIE2A
sts TIMSK2, r16 ; enable interrupt on compare match A
ser r16
```

```
out DDRD, r16            ; Port D output -- LCD segment cont
rol
clr r5
ldi r16, 0x55
mov r6, r16
clr r7
clr r8
ldi r16, 9
mov r10, r16
mov r11, r16
mov r12, r16
mov r13, r16
dec r10
    sei                          ; enable global interrupt
ldi r16, (1<<SE)
out SMCR, r16 ; use Idle mode here, waiting 1 second
clr r2
iniw:sleep
    dec r2
brne iniw
ldi r16, (1<<SM1)|(1<<SM0)|(1<<SE)
out SMCR, r16 ; use power-save mode
```

```
nop
nop
clr r2
loop:
nop
nop
sleep
nop
nop
dec r2
dec r2
breq adjtime
rjmp loop
adjtime:
ldi r17, 10
inc r10
cp r10, r17
brne updcount
clr r10
inc r11
cp r11, r17
brne updcount
```

```
clr r11
inc r12
cp r12, r17
brne updcount
clr r12
inc r13
cp r13, r17
brne updcount
clr r13
updcount:
    rcall calcor
rjmp loop
isr_timer2:
    clr r16
out DDRC, r16 ; float all COMx pins
bst r4, 1
brts show34
bst r4, 0
brts show2
mov r0, r5
ldi r18, 1
rjmp sel
```

```
show2:
    mov r0, r6
ldi r18, 1<<1
rjmp sel
show34:
    bst r4, 0
brts show4
mov r0, r7
ldi r18, 1<<2
rjmp sel
show4:
    mov r0, r8
ldi r18, 1<<3
sel:
    bst r4, 2
brtc lcden
    com r0
com r16
lcden:
    out PORTC, r16
out PORTD, r0
    out DDRC, r18
```

```
iext:inc r4
    reti
calcor:                ; translate R10~~R13 to R5~~R8
    clr r5
clr r6
clr r7
clr r8
    ldi ZH, high(table<<1)
ldi ZL, low(table<<1)
add ZL, r10
lpm ; load table data to R0
rcall filler
ldi ZL, low(table<<1)
add ZL, r11
lpm
rcall filler
ldi ZL, low(table<<1)
add ZL, r12
lpm
rcall filler
ldi ZL, low(table<<1)
add ZL, r13
```

```
lpm
rcall filler
ret
filler:
rol r0
rol r5
rol r0
rol r5
rol r0
rol r6
rol r0
rol r6
rol r0
rol r7
rol r0
rol r7
rol r0
rol r8
rol r0
rol r8
    ret
```

百家分析：此例中，中断程序使用了 inc r4，但没有保存 SREG，返

回时可能会使系统主程序工作不正常。但是实际上中断程序不会干扰主程序工作，因为主循环中有 sleep， CPU 进入休眠模式了，等待中断唤醒它，而在下一个中断到来之前早又开始休眠了。

这种笔划式液晶驱动比较简单，公共端（背极）接方波，笔划的另一端由单片机通过一个 2 输入异或门控制。异或门的一个输入端接背极的那个方波，另一端由单片机的 I/O 口控制。要显示某一段笔划时，使对应 I/O 口输出 1，不显示时输出 0。

北京中显电子有限公司祝你研发愉快！