

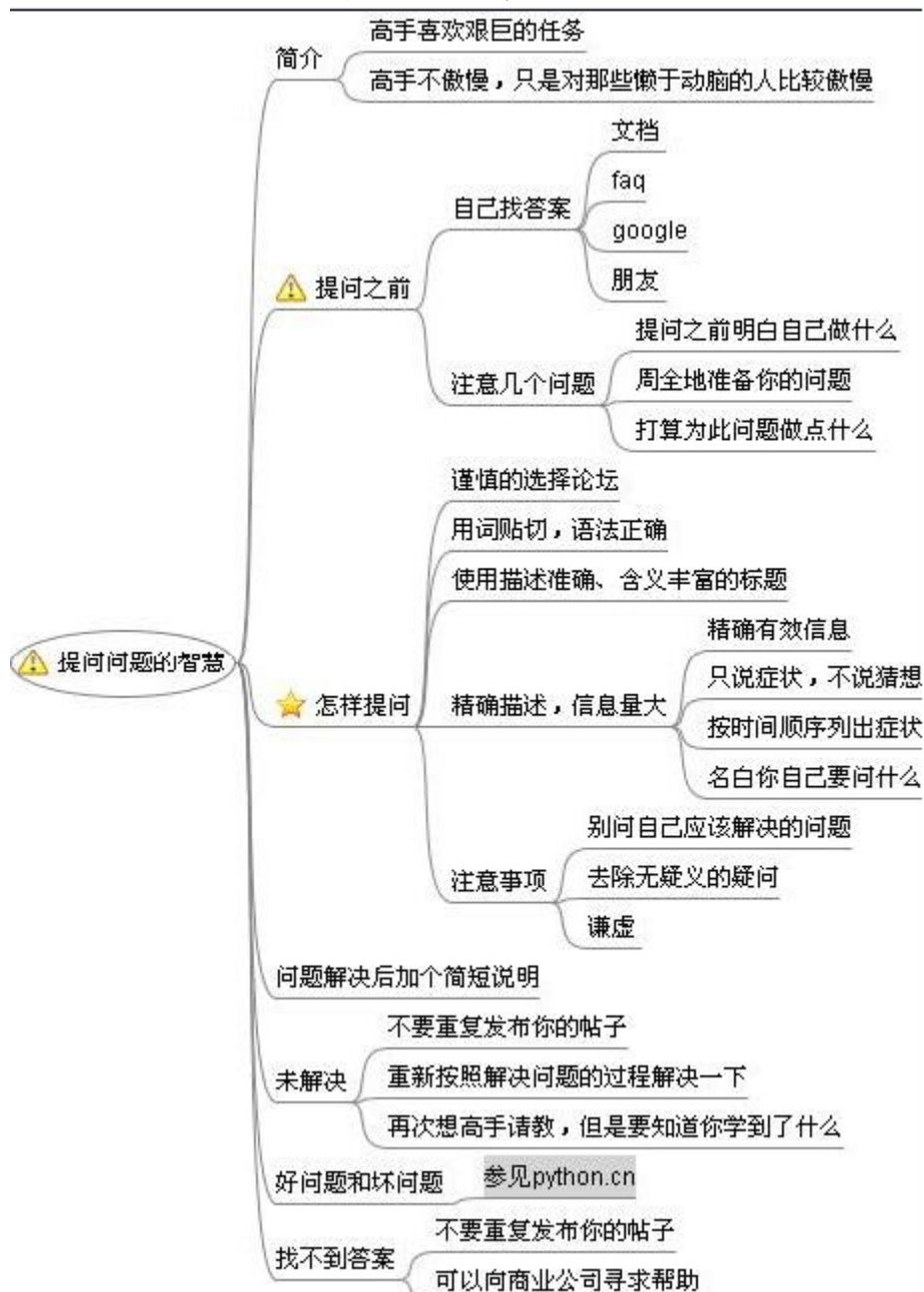
前言

由于这是本人第一次制作电子文档类的使用说明，我对一些截图软件，文档的排版都不熟悉，可能制作的东西显得并不整洁和清晰。

更由于本人所在的学校国家连技校学历也不承认，可以说是个冒牌的技校生，所以本人水平实在有限，文中一些问题的说明可能本身就有错误！还请大家理解。

本人单片机，51，AVR，C语言纯属自学，水平更是没的谈，所以大家不要对本文档的深度有太高的期望值，望见谅

提问的艺术

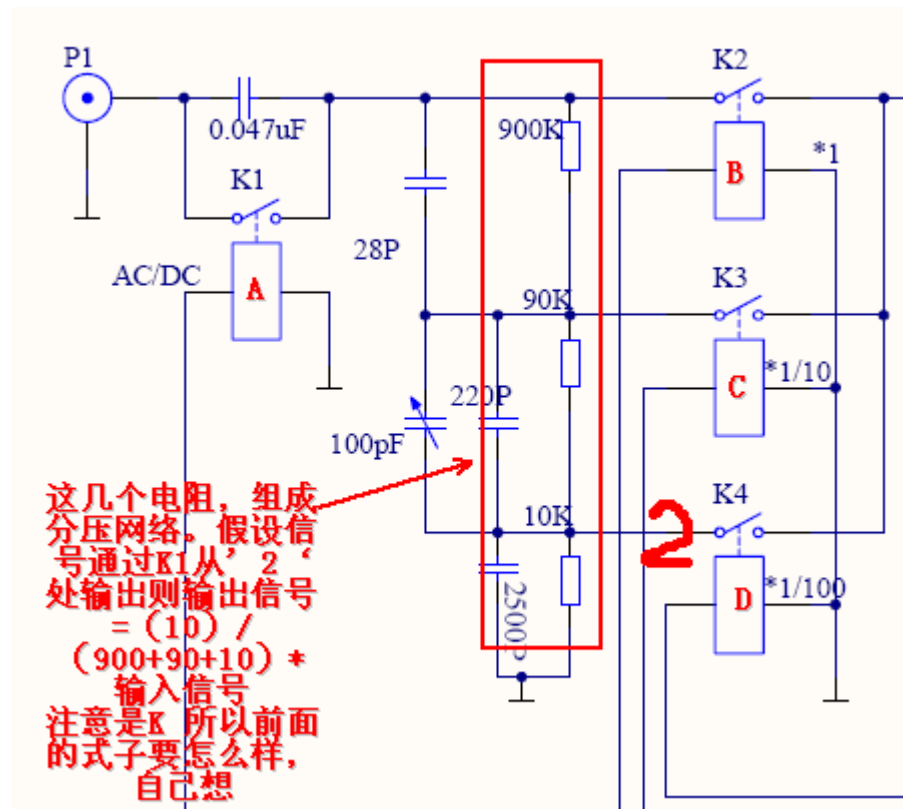


经常在学校有些人问我问题，有的人提的问题，我会思考回答。而有些人的问题，我则不屑回答！原因有二：其一这个问题他根本就对问题分析过，现在找个人带劳。其二：分析了问题，但是资料一个也没看，甚至就是书上有的参考！直接来问！啊 救命啊，遇到这样的事情，我直接让其回去看书，或者给他一个地方让他自己找资料看去。

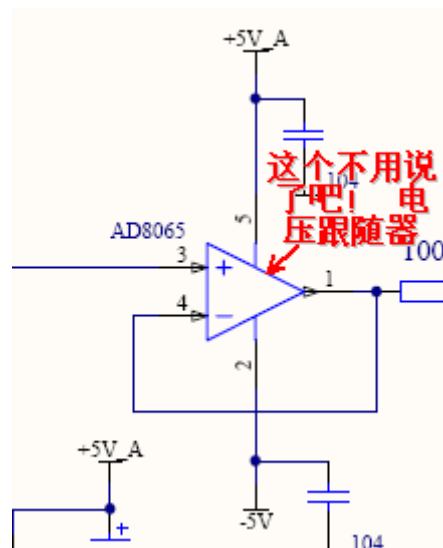
不是我傲慢而是我根本就无法回答，象怎么将十进制转为十六进制这样的愚蠢问题。书上有的问题，不好好看书 直接出来问人 。怎么能让人不反感！所以这样的问题 我拒绝！

回想起来上二年级了，除了正常的上课不懂的问题，我一定要问外，别的问题我还真没问一个。 因为学校有时要参加省里或者什么地方比赛，让谁去参加 竞争很激烈，但是我不怕，因为我从来不向辅导老师提问，从来不提一些基础问题，自己找答案。这样别人就无法知道你的深浅，就会器重你，相反一天到晚问这个问哪个的人，说明根本就没有真才实学，问的问题越多，暴露的问题越多，最后你只能淘汰出局。可以想象隐藏自己是多么的必要。自己不知道的东西，可以自己不知道，但是不能让别人知道你不知道。

呵呵，不说了。再说就要到伊拉克了。本来写了一部分就感觉到很疲倦。但是想到“古道热肠”网友要送我个 P3 听听，想到还有这么多的网友的期盼，我就有了精神！继续写！

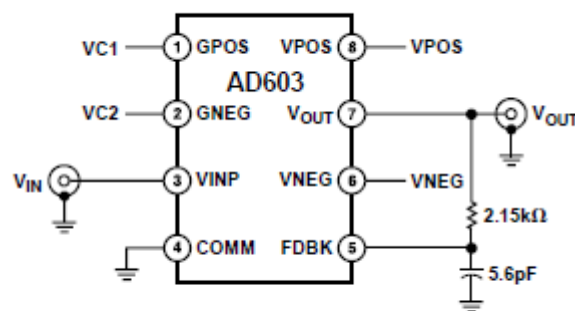


通过三个电阻的分压来达到衰减，输入信号的作用。继电器 B，C，D 起选择衰减倍数的作用。A 是选择 DC，或者 AC。

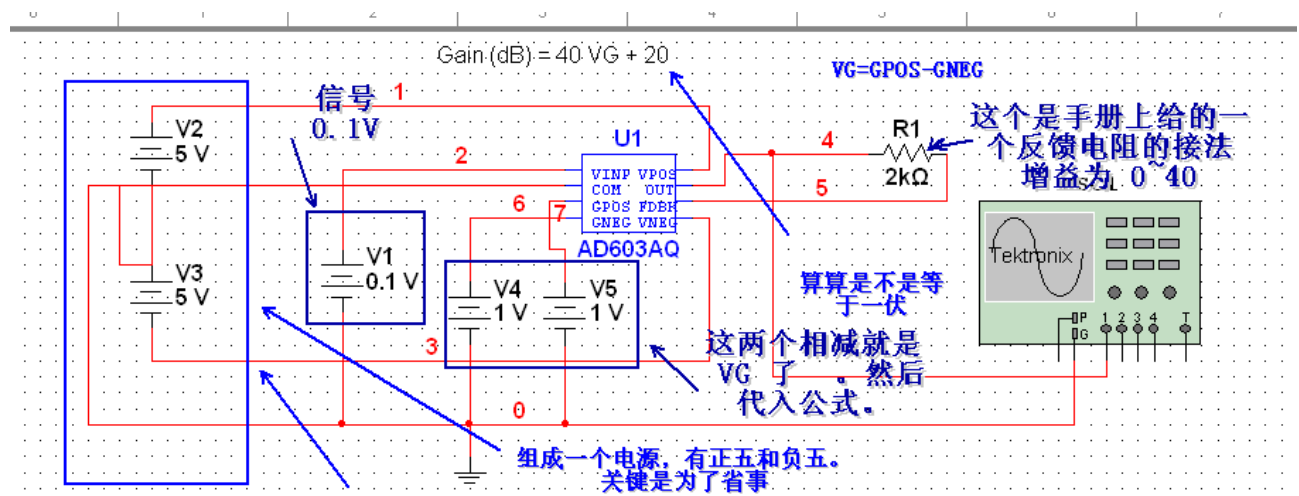


下图中 V2 和 V3 是起一个正五伏和负五伏的作用，是为了给 AD603 供电，图中所示接法可提供 0~40dB 的增益

a. -10 dB to +30 dB; 90 MHz Bandwidth



b. 0 dB to +40 dB; 30 MHz Bandwidth

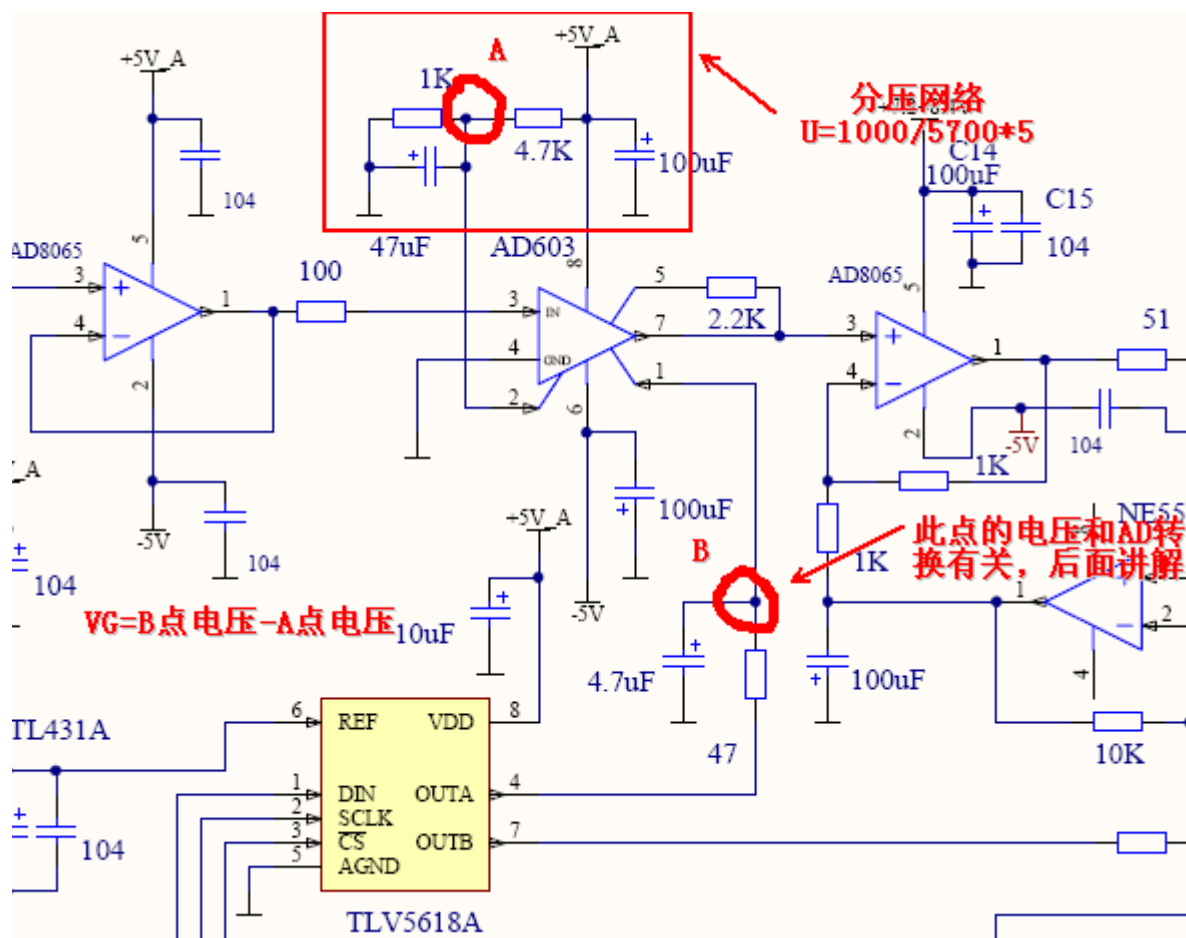


根据公式 VG 为 0 时的增益为 20，查表倍数为 10。V1=0.1V 则输出应该为 1V。

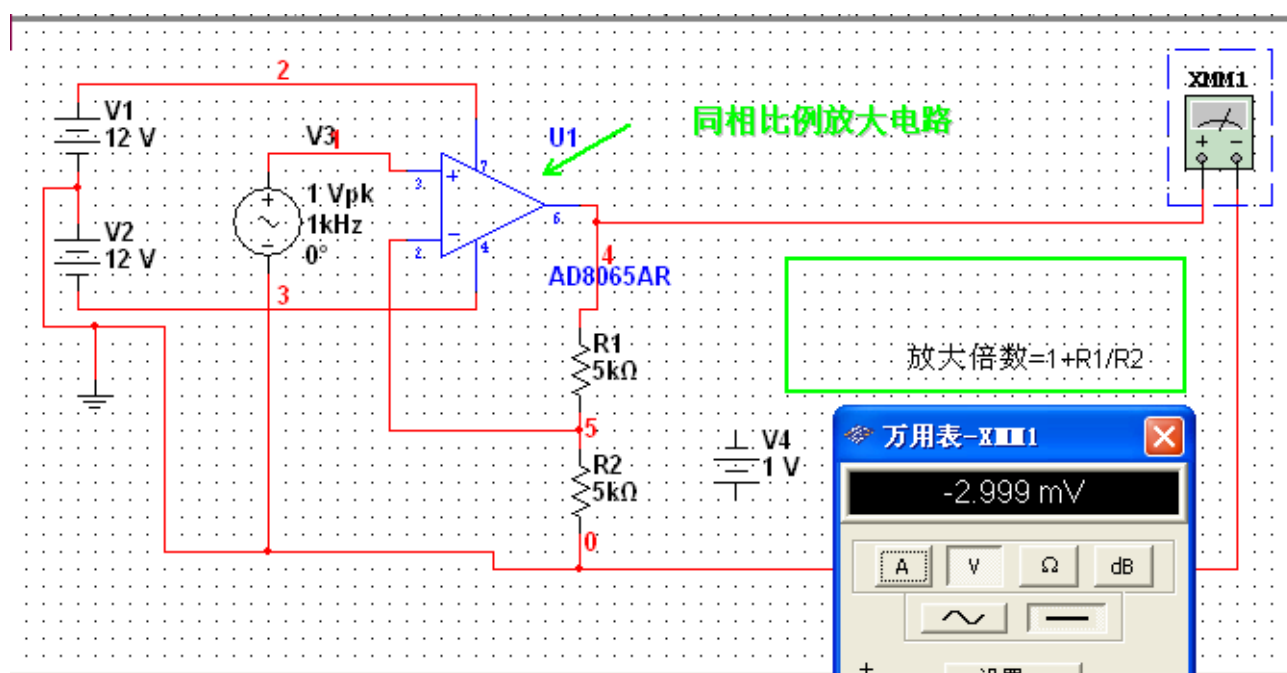
表 1-5 电压比和功率比的分贝表

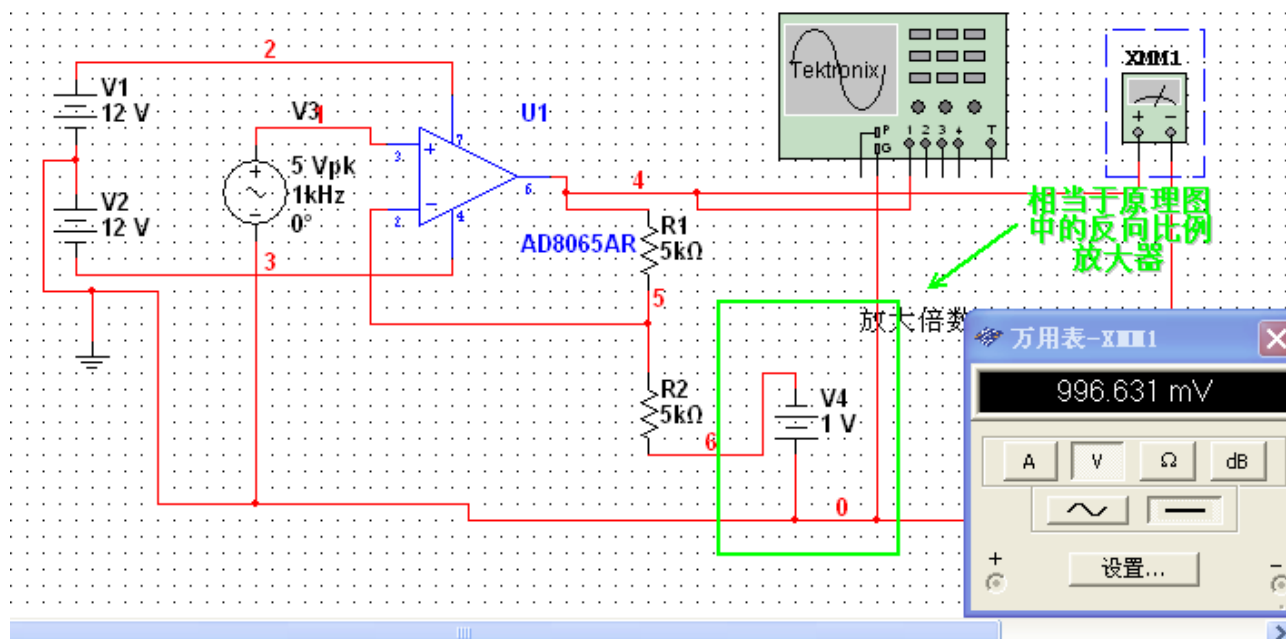
分贝	功率比	电压比
100	10000000000	100000
90	1000000000	31623
80	100000000	10000
70	10000000	3162
60	1000000	1000
50	100000	316.2
40	10000	100.0
30	1000	31.62
20	100	10.00
10	10	3.162
0	1	1.000
-10	0.1	0.3162
-20	0.01	0.1000
-30	0.001	0.03162
-40	0.0001	0.01000
-50	0.00001	0.003162
-60	0.000001	0.001000
-70	0.0000001	0.0003162
-80	0.00000001	0.0001000
-90	0.000000001	0.00003162
-100	0.0000000001	0.00001000
10	10.0000	3.1623
9	7.9433	2.8184
8	6.3096	2.5119
7	5.0119	2.2387

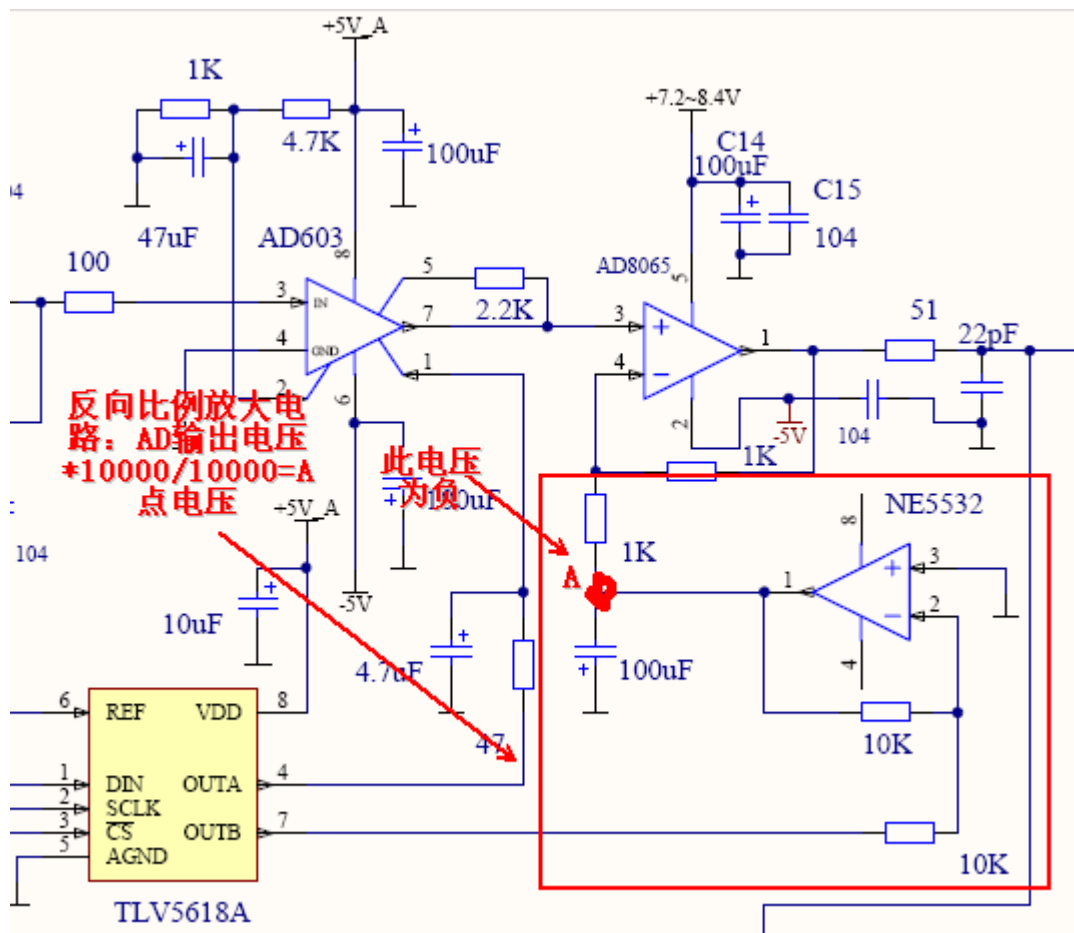




看到这里，我想大家明白了 AD603 的工作原理，既通过 TLV5618 来改变 B 点的电压来改变 VG 进而改变 AD603 的增益。







解释下：通过 TLV5618 的输出电压经过 NE5532 组成的反向器把正相电压转换为等比例的负电压，这个负压接到 AD8065 的反向输入端，大家可以通过这个电压来确定 AD8065 的输出，反映到屏幕上就应该是波形的中线位置。如果你的东西显示的波形偏下你可以增大 TLV5618 的 B 端输出来调节。

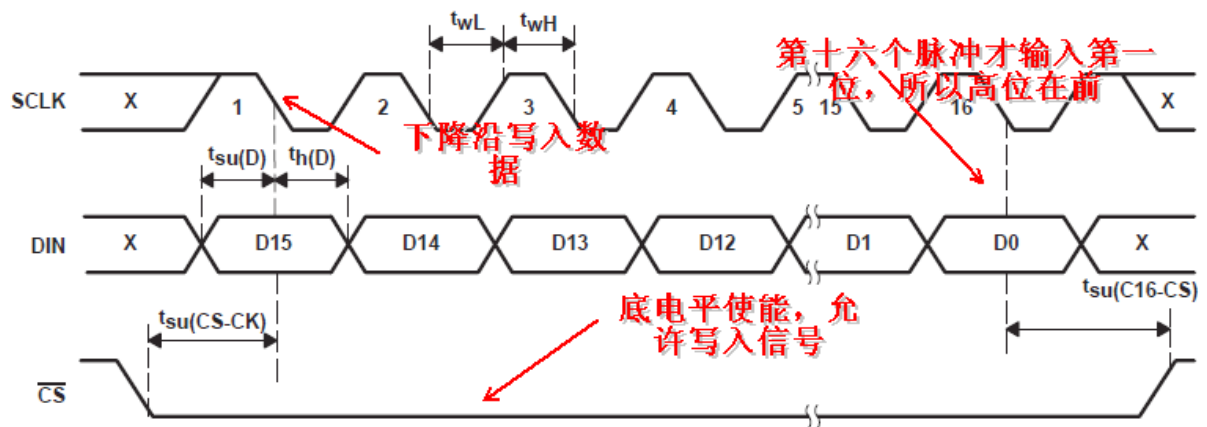


Figure 1. Timing Diagram

```
#include <mega16.h>
#include <delay.h>
void intda() //初始化
{
    #define DA_CE PORTC.5 //定义功能引脚    使能端
    #define DA_CLK PORTC.6 //时钟引脚
    #define DA_DAT PORTC.7 //数据引脚
    DDRC|=0XE0;           //定义为输出
    DA_CE=1;              //开始前 全部置高
    DA_DAT=1;
    DA_CLK=1;
}
```

```
void spi_out_a(unsigned int j,unsigned char q)
{
    unsigned char u;
    j=j&0xfff; //由于只有十二位，所以把前面的数去掉
    if(q=1) //判断是给 输出 A 的数据，还是 B 的数据
    {
```

register-select bits

R1	R0	REGISTER
0	0	Write data to DAC B and BUFFER
0	1	Write data to BUFFER
1	0	Write data to DAC A and update DAC B with BUFFER content
1	1	Reserved

这一位为0，则是B的

为1则，是A的

```
j=j+0x8000;
```

```

}
DA_CE=1;
DA_DAT=1;
DA_CLK=1;
delay_us(1);//延时
DA_CE=0;//使能底了，现在可以写数据了
for(u=0;u<16;u++)//循环 16 次
{
    if(j&0x8000)//这一位为高，则执行下面的语句，即 写 1。否则
    {
        DA_DAT=1;
    }
    else
    {
        DA_DAT=0;
    }
    delay_us(1);//延时
    DA_CLK=1;//时钟置高
    delay_us(1);
    DA_CLK=0;//置低，也就是说，制造了个下降沿，这个时候信
//号已 经 写 入 了
    delay_us(1);
    j<<=1;//左移动一位
}
    delay_us(1);
    DA_CE=1;//16 位都写完了，势能禁止
}

void main()
{

unsigned int j=0;
unsigned int k=0;
intda();//初始化

while(1)
{

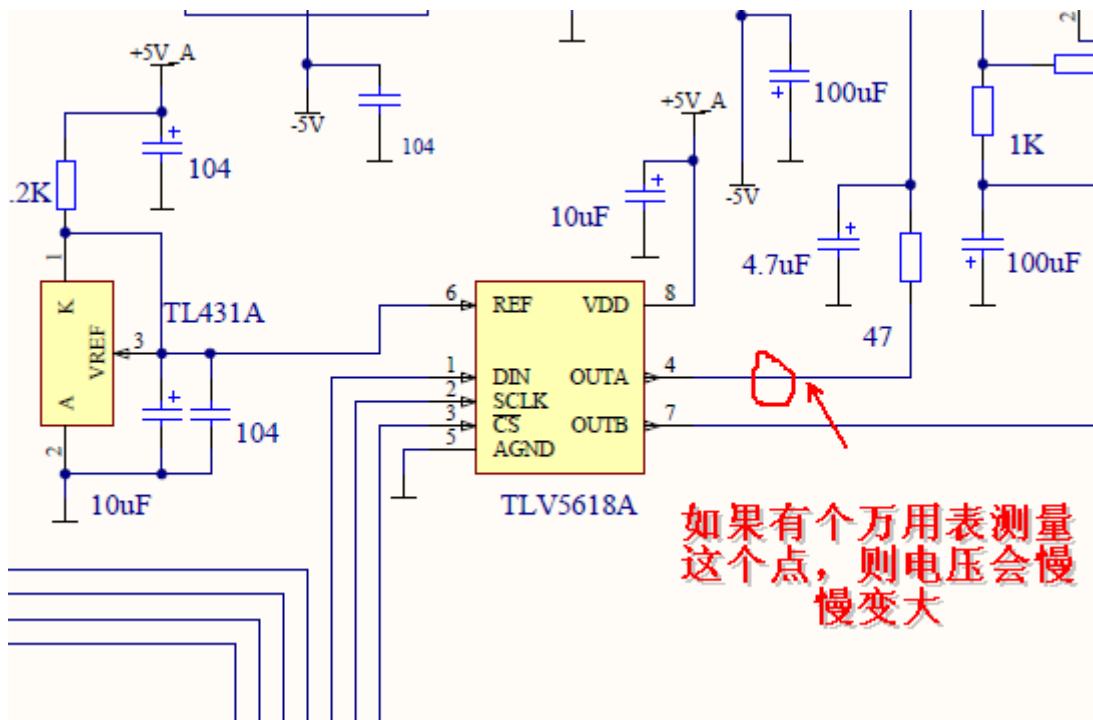
```

```

spi_out_a(k,1);    //将 K 这个数据，写入 A
    k=(k<4095)?k+1:0; //
    delay_ms(10);
}

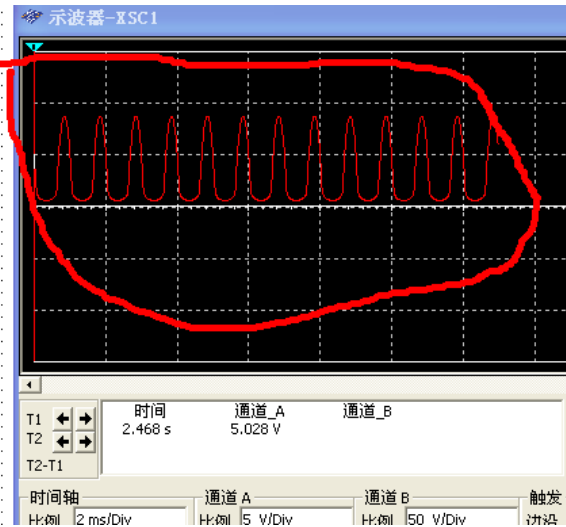
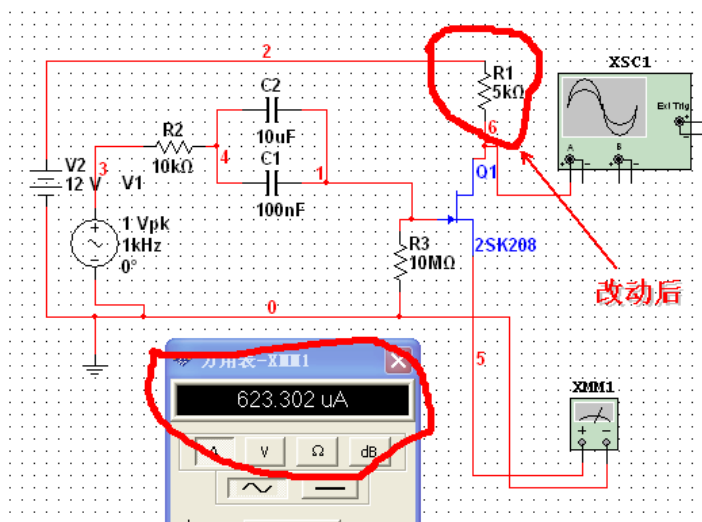
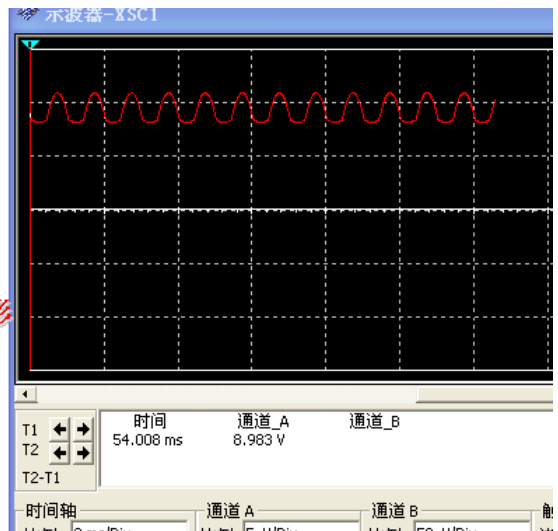
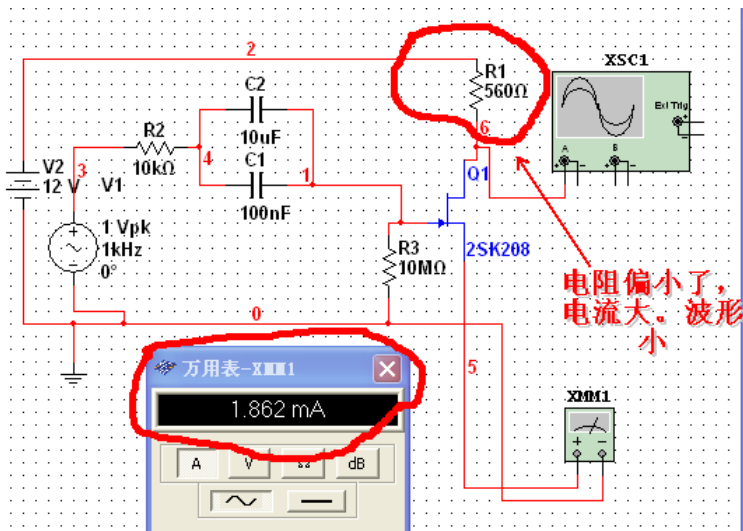
}

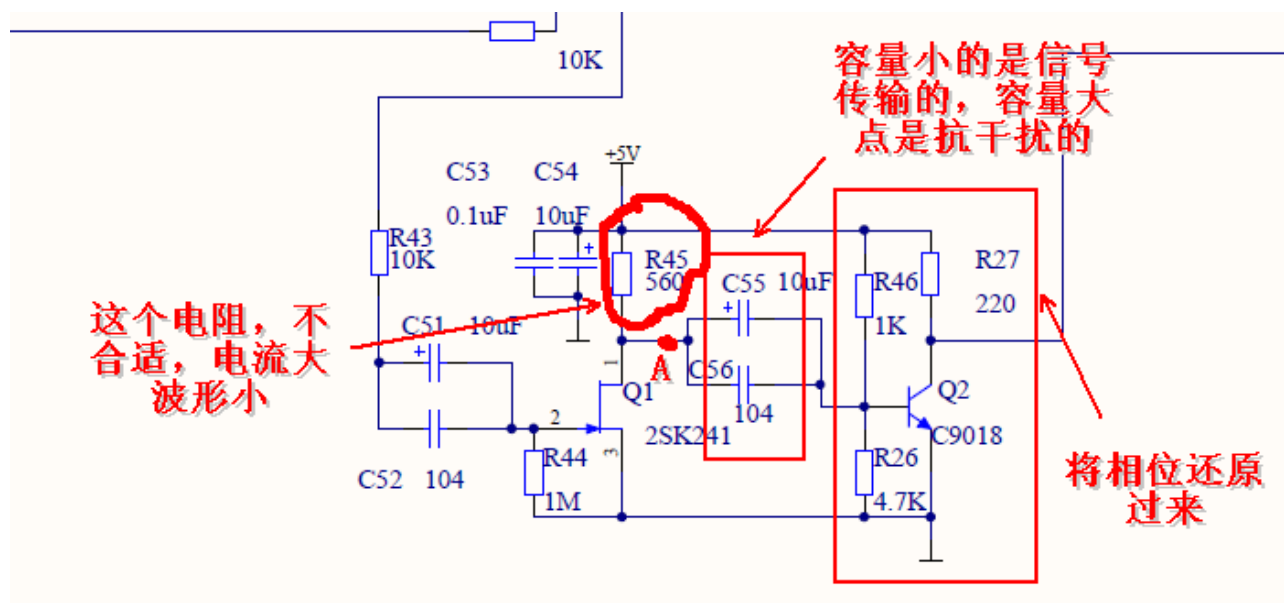
```



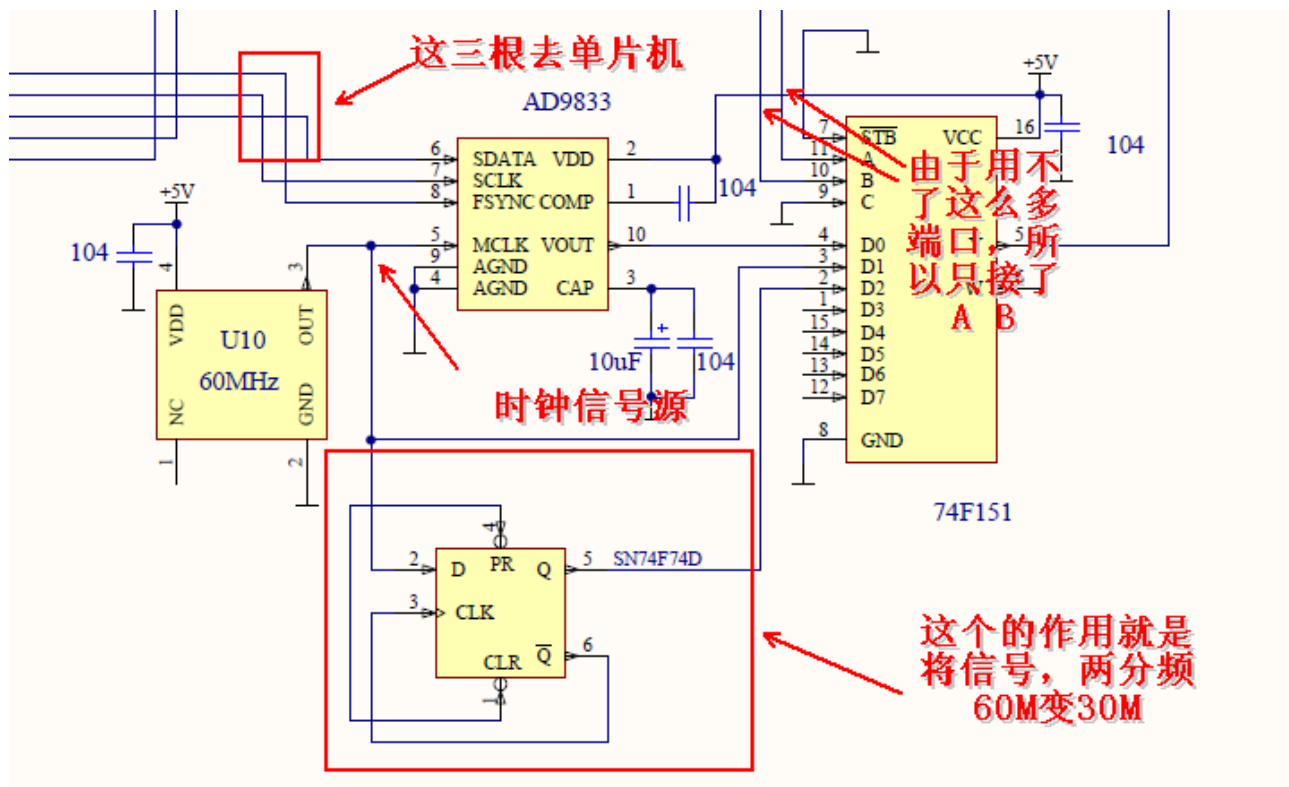
当然也可以写 B 点的，只要将 `spi_out_a(k,1);` 中的 1 改为 0 即可。

当然这个不是原版程序中的驱动，而是我自己写的一个，大家仅供参考好了，大家可以方便的通过这个程序来控制 DA 的输出电压，也就是说，可以方便的改变 AD603 的增益，和输出电压的大小。





本人认为，着部分电路是起的将，信号的脉冲提取出来，送到MCU2即MEGA128的时钟输入端，来统计，进而算出频率。着一部分电路决定了记的频率是否准确。这一部分电路我认为，设计的不理想，首先有一个电阻选的有些小，电流浪费了，波形还取的小。将相位还原出来着部分电路完全可以省掉，既然是测量频率就没必要将频率还原到原来的相位，只要有个下降沿就可以了，故此本人认为相位还原这部分电路可以省去。我认为因该，改为电压比较器，超过一定的电压后的脉冲才输出一个脉冲。以上只是我的一些浅见。



AD9833 的引脚

引脚功能描述

引脚号	名称	功能
电源		
2	VDD	提供给模拟和数据接口部分的正电源，片内2.5 V稳压器也由VDD供电，VDD可为2.3 V到5.5 V。在VDD 和 AGND之间必须接一个0.1 μF 和一个10 μF的去耦电容。
3	CAP/2.5 V	数字部分由2.5 V电源供电，这个2.5 V电源由片内的稳压器(当VDD超过2.7 V时)从VDD产生。这个稳压器需要一个典型为100 nF的、连接于CAP/2.5 V 和DGND之间的去耦电容。如果VDD等于或小于2.7 V时，CAP/2.5 V应直接接到VDD。
4	DGND	数字地
9	AGND	模拟地
模拟信号及参考		
1	COMP	DAC的偏置脚，该引脚用于给DAC的偏置电压去耦。
10	VOUT	电压输出。AD9833的模拟和数字输出由该引脚提供。不要求外接负载电阻，因为片内有一个200 Ω的电阻。
数字接口和控制		
5	MCLK	数字时钟输入。DDS的输出频率取决于MCLK频率的二进制分式。输出频率的准确度和相位噪声由该时钟决定。
6	SDATA	串行数据输入。16-bit的串行数据字加到该引脚上。
7	SCLK	串行时钟输入。数据在每个SCLK时钟的下降沿被送入AD9833
8	FSYNC	低使能控制输入。这是输入数据的帧同步信号。当FSYNC低时，通知内部逻辑有新数据送入器件。

9833控制寄存器

D15 D14	D13 B28	D12 HLB	D11 FSELECT	D10 PSELECT	D9 保留	D8 RESET	D7 SLEEP1	D6 SLEEP2	D5 OPBITEN	D4 保留	D3 DIV2	D2 保留	D1 MODE	D0 保留
00: 写控制	0: 写14位	0: 写低 1: 写高	0: 用FREQ0	0: 用FREQ0	0	0: 取消复位	0: MCLK 工作	0: DAC 工作	0: 输出DAC	0	×	0	0: 正弦波 1: 三角波	0
	1: 写28位 先低后高	×	1: 用FREQ1	1: 用FREQ1		1: 9833 复位	1: MCLK 禁止	1: DAC 禁止	1: 输出MBS		0: MBS/2 1: MBS			
01: 写FREQ0	FREQ0 寄存器的位													
10: 写FREQ1	FREQ1 寄存器的位													
11: 写相位	0: 写PHASE0	×	MSB			PHASE0 寄存器的位								
	1: 写PHASE1	×	MSB			PHASE1 寄存器的位								

Figure 2. Master Clock

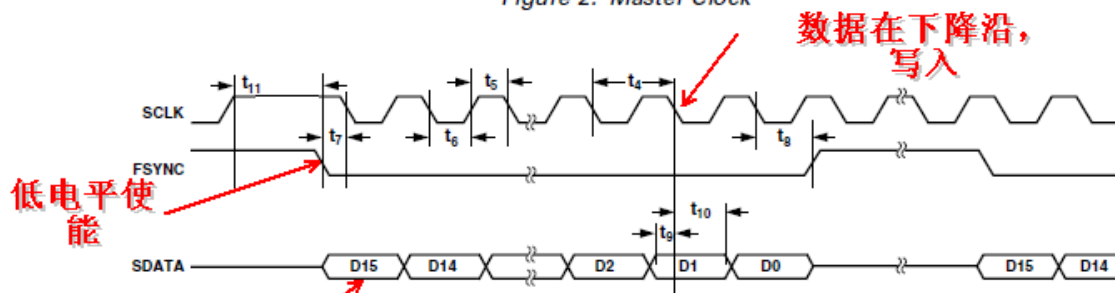


Figure 3. Serial Timing

高位先 输入

看程序前先讲一下 AD9833 的输出频率公式：

$$f = \Delta \text{Phase} \times f_{\text{MCLK}} / 2^{28}$$

用文字表述就是：

输出频率=写入的数据值*有源晶体的震荡频率/268435454

```
void write_2byte(unsigned int a)//这个函数的作用是，数据写入
{
    unsigned char i;
    DDS_CLK=1;//开始前先把时钟置高
    DDS_DAT=1;//开始前先把数据置高
    DDS_CE=1;//开始前禁止使能
    delay_us(2);
    DDS_CE=0;//使能开了，这个时候可以写数据了
    for(i=0;i<16;i++)
```

```

    {
        if(a&0x8000)//先写的高位，
        {
            DDS_DAT=1;
        }
        else
        {
            DDS_DAT=0;
        }
        DDS_CLK=0;//制造一个下降沿把数据写入
        DDS_CLK=1;
        a<<=1;//左移动一位
    }
    DDS_CE=1;//完工了，使能拉高
    DDS_CLK=0;
}

```

```

void init_dds(void)
{
    write_2byte(0x2100);//大家可以对照 9833 控制器，对号入座
    write_2byte(0x2000);//取消复位
    write_2byte(0x4000);
    write_2byte(0x403f);
    write_2byte(0x2900);
    write_2byte(0x2100);
    write_2byte(0x8000);
    write_2byte(0x803f);
    write_2byte(0xc000);
    write_2byte(0x2000);
}

```

```

void DDS_out(unsigned long freq_value)
{
    unsigned long dds;
    unsigned int dds1,dds2;
    dds=freq_value*8.94784853333336;//很多人看不懂这个数据，其实//

```

//大家可以完全用，我上面给出的这个公式来算，既然大家想知道，我就告诉大家，这个值其实是所求数据与实际数据的一个比值。大家也可以方便的算出来，显然这里带上这样一个比值要比用公式算要好的多，带上这个数据算和用公式算出的结果是一样的。可能是公式中数据太常处理上很不方便，所以引用了这个比值。可是结果是导致很多人没法通过看程序看出什么意识。下面的程序就简单了有效数据是 28 位，还有几位是选择写在什么地方的了，大家可以参照 9833 控制器

```

dds=dds<<2;
dds1=dds;
dds2=dds>>16;
dds1=dds1>>2;
dds2=dds2&0x7fff;
dds2=dds2|0x4000;
dds1=dds1&0x7fff;
dds1=dds1|0x4000;
write_2byte(0x2028);
write_2byte(dds1);
write_2byte(dds2);
}

```

151 的的输出逻辑图

Inputs				Outputs	
Select			Strobe S	Y	W
C	B	A			
0	0	0	H	L	H
L	L	L	L	D0	D0
L	L	H	L	D1	D1
L	H	L	L	D2	D2
0	0	1	L	D3	D3
H	L	L	L	D4	D4
H	L	H	L	D5	D5
H	H	L	L	D6	D6
H	H	H	L	D7	D7

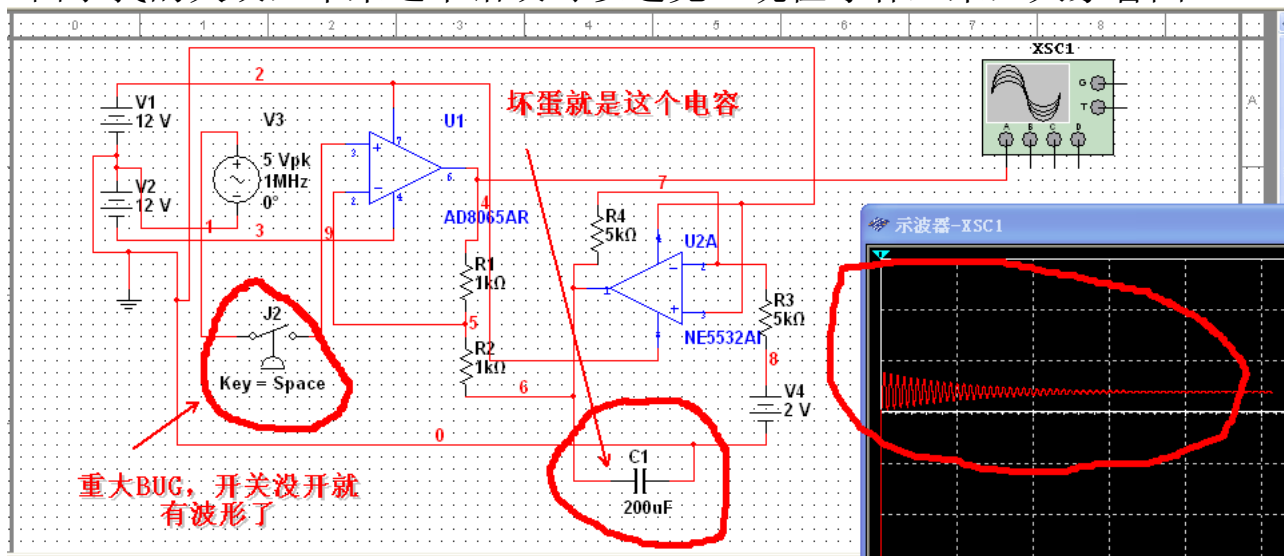
H = High Level, L = Low Level, X = Don't Care
D0, D1...D7 = the level of the respective D input

二进制数是多少就选择哪个叫的信号输出

由于使能端已经在电路中接低电平，所以才程序中不用考虑。图中：比如，C，B 端是低电平 A 点是高 对应二进制为 1，所以选择的是 D1 端的信号到 Y。（其实写这个很没必要，如果这个也看不懂。其实就没必

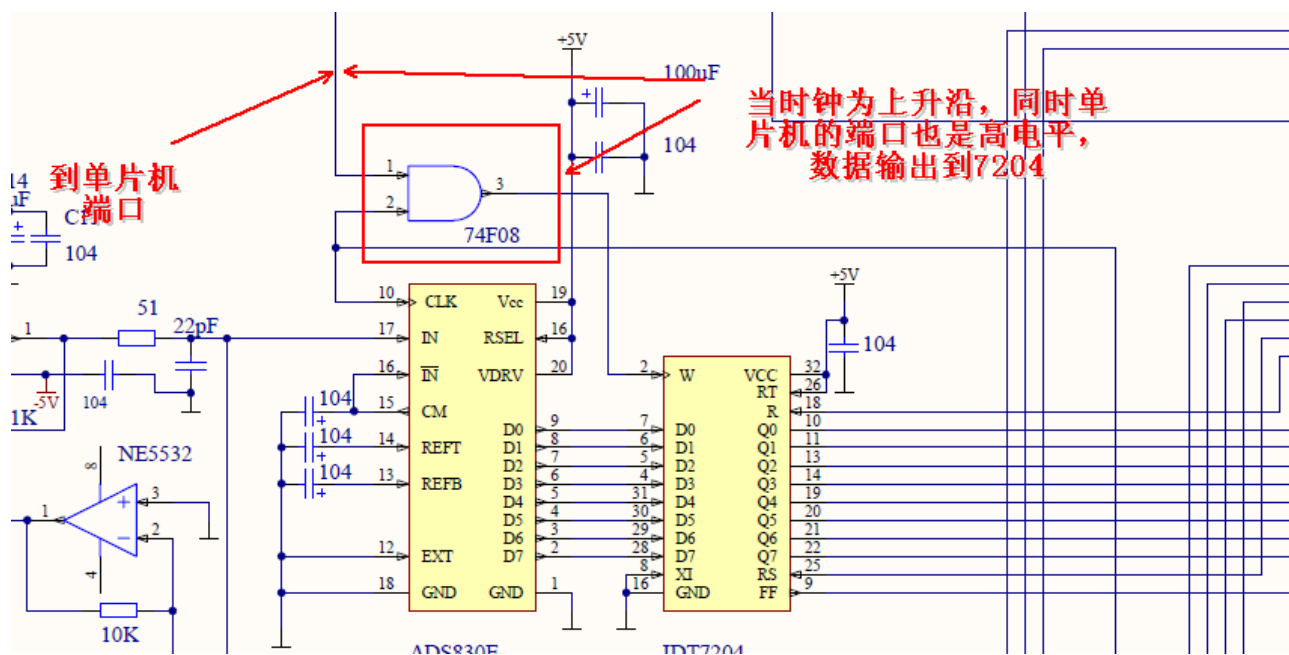
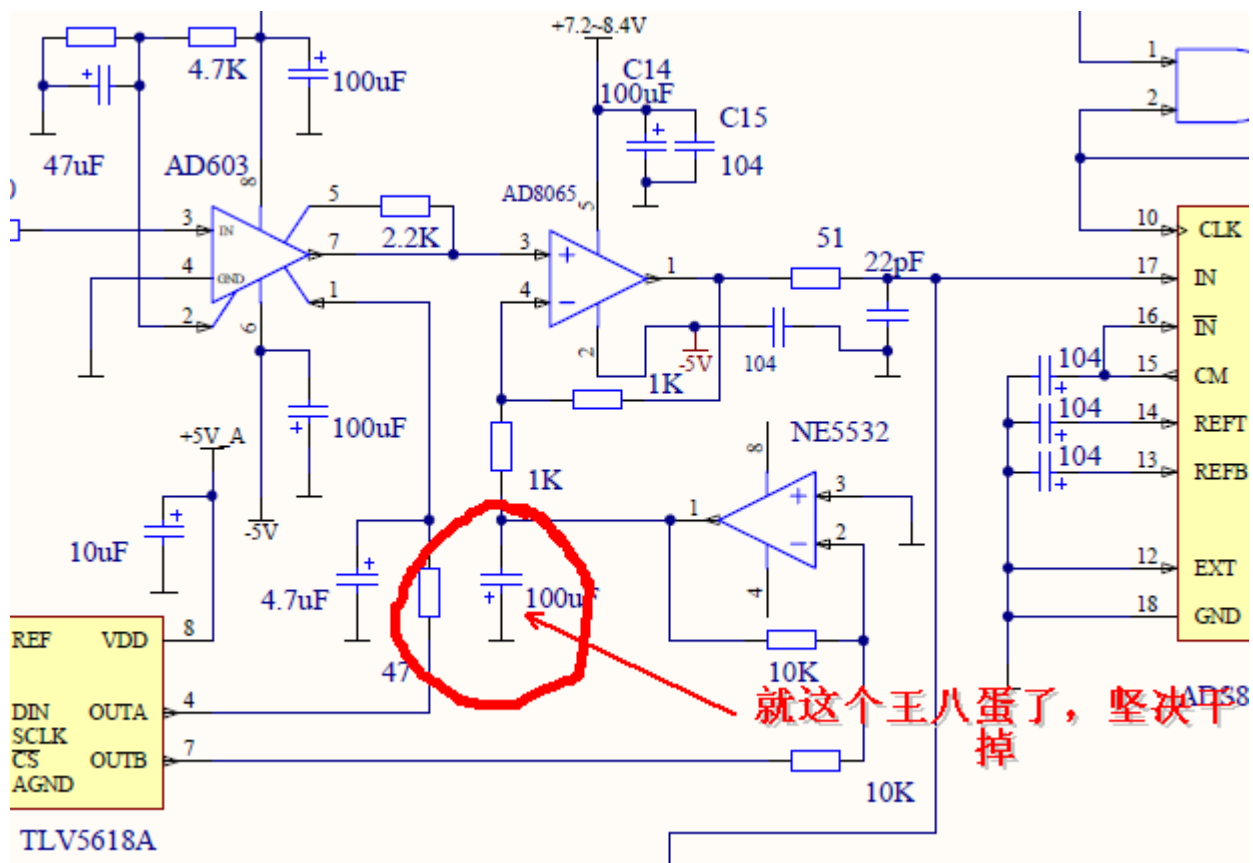
要看下去了，趁早换行，免得耽误你的前程：)

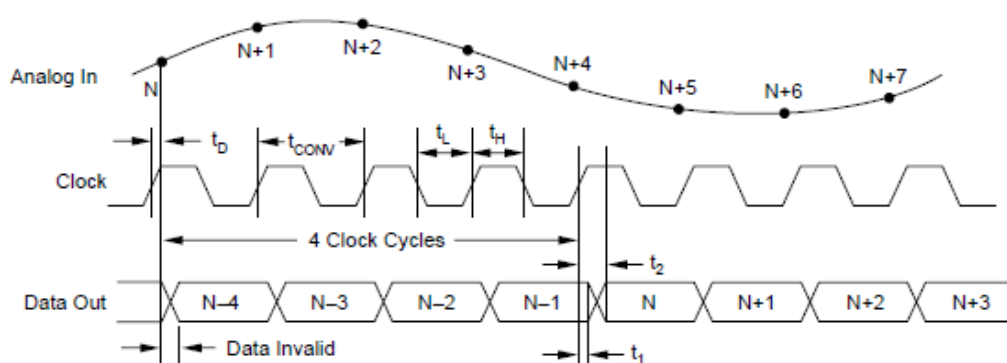
由于我的失误，本来这个错误可以避免。现在才补上来，大家看图



本来这个图早就截了，我也想到这个问题，也仿真了，建议大家去掉这个电容，但是仿真并没有发现问题。也就放弃了。后有网友提出有这一问题，分析还是通不过，所以再次仿真，发现BUG。可以说很严重！本来上一版是建议大家去掉这个电容，这次是坚决初掉这个出力不讨好的家伙！！

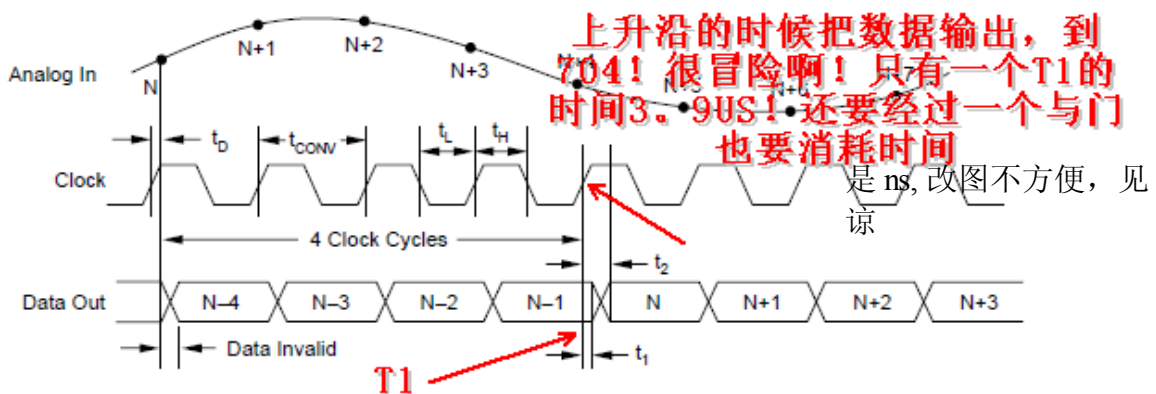
如果不去掉也就是说每次，开机。或者不输入信号你的显示屏也会有波形。去不去掉不用我说了吧





SYMBOL	DESCRIPTION	MIN	TYP	MAX	UNITS
t_{CONV}	Convert Clock Period	16.6		100 μ s	ns
t_L	Clock Pulse Low	7.3	8.3		ns
t_H	Clock Pulse High	7.3	8.3		ns
t_D	Aperture Delay		3		ns
t_1	Data Hold Time, $C_L = 0pF$	3.9			ns
t_2	New Data Delay Time, $C_L = 15pF$ max		5.9	12	ns

将高速数模转换器 ADS830E 的工作简单介绍一下，由图可知每个时钟周期进行一次数模转换，所以采样速率就是时钟频率，故可以很方便的通过控制采样时钟来控制采样频率，当前输出的采样数据是 4 个时钟周期以前采样电压的值，也就是说从采样到输出有 4 个时钟周期的延迟，这对我们所要做的电路并不重要，所以我们可以简单的理解为输入一个时钟脉冲转换一次，时钟的脉冲的下降沿输出数据就行，应用非常方便。还有一点就是 ADS830E 的输入电压幅度是可以编程控制的，11 脚（RSEL）为控制引脚，当 11 脚置高电平时，ADS830E 的输入电压范围是 1.5V~3.5V，即 2V_{pp}。当 11 脚置低电平时，输入电压范围是 2V~3V，即 1V_{pp}。



SYMBOL	DESCRIPTION	MIN	TYP	MAX	UNITS
t_{CONV}	Convert Clock Period	16.6		100 μ s	ns
t_L	Clock Pulse Low	7.3	8.3		ns
t_H	Clock Pulse High	7.3	8.3		ns
t_D	Aperture Delay		3		ns
t_1	Data Hold Time, $C_L = 0pF$	3.9			ns
t_2	New Data Delay Time, $C_L = 15pF$ max		5.9	12	ns

为什么说冒险，因为数据在过了 T1 的时候，刚好处于数据的变化区，这个数据是不准的。由于 7204 是上升沿读入数据，（即 7204 的 2 脚 W）他通过一个与门，将时钟信号和单片机的信号相与，如果同为高，则写入数据，可是这个与门也耽误时间，这样数据的安全就受到了威胁。

关于 LCD 的低闪烁来讲，闪烁并不是把数据在某一点上写本身，而是在清屏与写屏间的明显闪烁，或者就是按键处理中的闪烁。这个电路本身按键采用的是中断处理只要中断时间处理的够快速就可以避免。但是在这个电路中中断处理起来就不是那么理想，因为在按键后还要改变屏幕上的东西，这个就耗时间了，如果在波形显示中来了按键，屏幕当然会一卡，解决的方法有点类，把长的任务，分为若干个小任务。比如在按键后要显示五个字母，那么 可以采用标记，写一个字母后返回主任务 连续五次，把一个卡，变为轻微的闪。还有就是在刷屏的时候不一定表格线每刷完一次屏就重新画一次，可以是刷了几屏后清了再写一次。

还有一个地方就是，7204 读取数据到单片机，所耽搁的时间。我认为 7204 的数据是很快写满的，可以在刷屏的间隙抽空把数据读出来，少量多次 。

以上只是我的一些浅见，本来不想将这个的。因为本身能力不足啊，纯属班门弄斧。

下面来写 LCD 的控制驱动

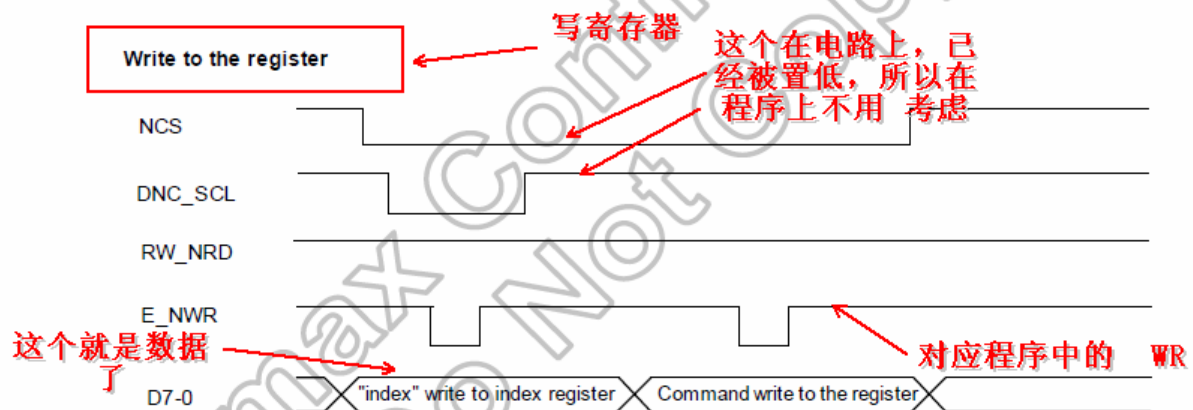
Table 5. 4 Data Pin Function for I80 Series CPU

Operations	E_NWR	RW_NRD	DNC_SCL
Writes Indexes into IR	1	0	0
Reads internal status	1	1	0
Writes command into register or data into GRAM	1	0	1
Reads command from register or data from GRAM	1	1	1

Table 5. 5 Data Pin Function for M68 Series CPU

寄存器的阅读指令

写指令进入寄存器
或者写数据



```
void WMLCDCOM(unsigned char data_1,unsigned char data_2)
```

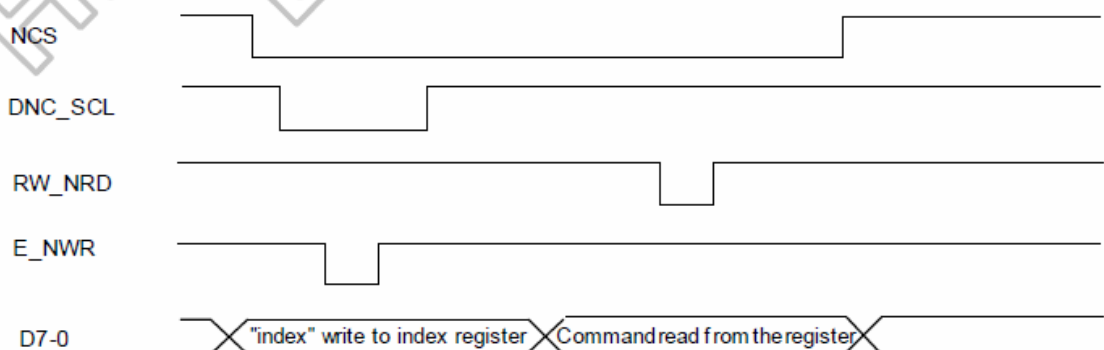
```
{
```

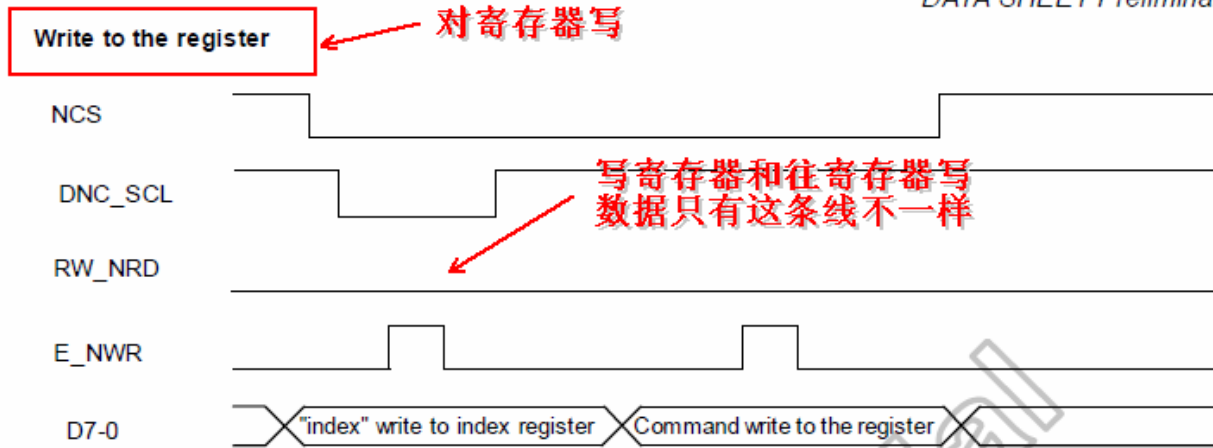
```
    RS=0;  
    PORTF=data_1;  
    PORTA=data_2;  
    WR=0;  
    WR=1;
```

```
}
```

Read the register

读寄存器





```
void WMLCDDATA(unsigned char data_1,unsigned char data_2)
{
    RS=1; //就是□ 个不□□,□ 个□ 高□ 个□ 低
    PORTF=data_1;
    PORTA=data_2;
    WR=0;
    WR=1;
}
```

先到□ 里