我不是为了做一个示波器

讲一个故事：

今年九月，一个新学期的开始，课很少。

我是一个闲不住的人，这样的日子很难熬，想去电子市场逛逛，但学校离市区有三十多公里路，终于无聊到周末了。

和平常一样，逛电子市场都是这儿看看哪儿看看，碰着没见过的还喜欢问问，多年的习惯改不掉的……

一家柜台上摆着“低价处理 LCD 模块”的牌子，对于像我这样的穷学生来说，价格往往是考虑的主要因素。我径直走了过去，老板说这些低价屏都是全新的，只是没有资料，所以只能低价处理，于是我就贪了个小便宜花 30 块钱买了一块 128*64 分辨率的点阵屏。喜欢贪小便宜的人最后往往都是要吃亏的，最后我真“吃亏”了，就因为这个屏，害得我花了近 300 块买了一块 320*240 的屏。

回到学校后就上网找它的资料，功夫不负有心人，我找着了。从资料中得知这块液晶显示器是不带字库的，这让我有些小失望，但一想只花了 30 块钱也就没事了。根据资料编写了程序让它显示了一些简单的图形，但让它显示图形或字符都得将所要显示的东西做成点阵数据存放在数组里才行，太浪费单片机里少得可怜的资源了！没有字库的点阵屏就是鸡肋！

正徘徊在“食之无肉，弃之有味”的时候，突然灵光一现，何不用它来显示一些时实的图形呢？显示什么呀？亮着的示波器给了我灵感，那就让它显示波形吧！正好我用的 AVR 单片机带有 AD 转换器，说干就干，忙活了一下午，晚上的时候波形就显示出来了。这不就是一个最基本的数字示波器吗？图 1 和图 2 就是当时的“珍贵照片”，因为那块电路在以后的试验中已经被我拆了。这两张照片是我买了 LCD 屏的第二天晚上照的。

第一步的成功，坚定了我做数字示波器的信念。人总是有追求的，所以我要完善它！其实这无异于“因有一只鞍而买一匹马”，但那并不总是坏的。

接下来的一个星期中我有事做了……

第二个周末，我拿出了我半个月的生活费，花三百多块买了一块 320*240 的液晶显示屏和两片 TLC5510，开始了我自制数字示波器的征程。

以前从没有想过要制作一台数字示波器，所以对数字示波器的了解也仅停留在基本原理和功能上，更深入的东西几乎是一片空白。但是我相信电类的知识，只要你下工夫就没有

学不来的，于是我开始到处找资料，但不管是图书馆还是互联网，几乎找不着一个现成的电路或实例可供参考，这种情况下只能自力更生，按自己的理解画出框图，一步一步的实验。

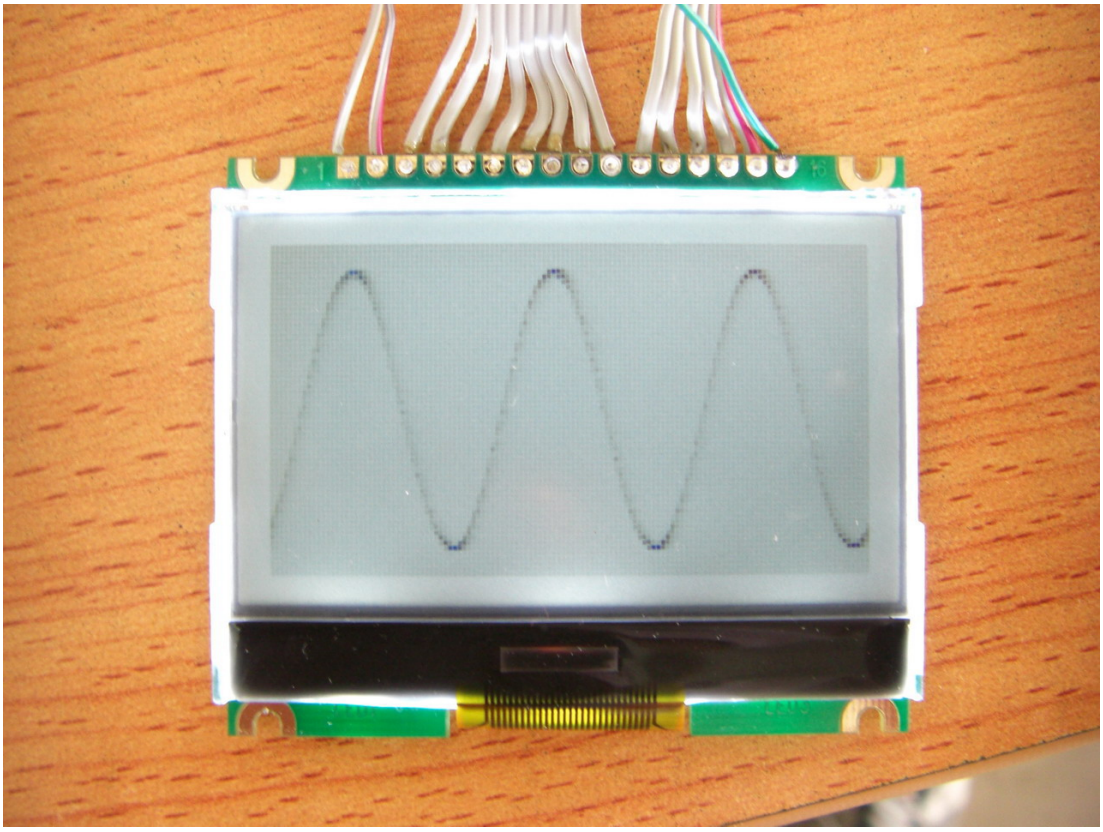


图 1：测量 1000Hz 的正弦波

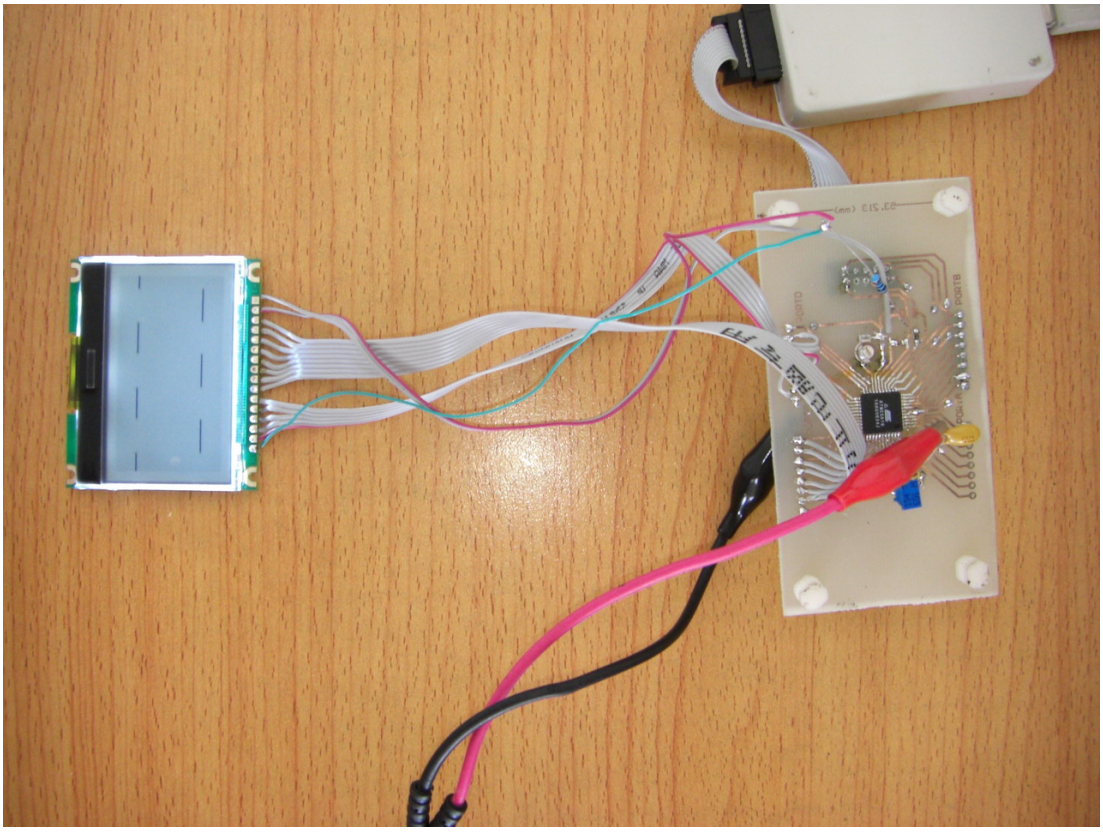


图 2：最初的“数字示波器”

刚开始时因为用的是 AVR 单片机中的 ADC，最快的转换速率为 40k（不要求达到 10 位精度），所以勉强能测到两三千赫兹，这样的性能我当然不满意！于是就用外置的高速 ADC，这时 TLC5510 就派上用场了，它的最高转换速率为 20MSa/s，为了达到高速采样我给它提供 20MHz 的时钟信号，8 位并行数据直接送入单片机，问题出来了，单片机太慢了，来不及读数据。通过查阅资料我了解到对高速数据的缓冲可以使用 FIFO 存储器，这是我第一次接触 FIFO 存储器。第三个周末，我花了 40 块钱买了一片 4KB 容量的 FIFO 存储器 IDT7204，直到现在这个示波器还是用的这片存储器。FIFO 的使用解决了单片机来不及读数据的问题，但剩下问题还有很多：如何控制扫速、如何控制灵敏度、如何控制触发……

太多的“如何”让我好几个晚上不知如何睡好觉，但也正是这些“如何”给了我解决掉这些“如何”的动力……我喜欢挑战自己。换方案，试验，调试，再换方案，再实验，再调试……最终一个个“如何”的被我解决掉了，兴奋又使我好几个晚上不知如何入眠。这就跟一个国家的内战一样，不管谁输谁赢受伤的总是这个国家，但却推动了历史的进步。我就是在发现问题和解决问题中进步的，虽然受了点伤……试验期间我得到了《无线电》杂志社尹飞编辑的鼎立帮助，非常感谢他在精神以及物质上对我的支持！

最终我将这个数字示波器做出来了，也实现了我起初的要求，能较好的测量到 5MHz，这样的性能与专业的示波器相比也许差了些，但用于音频电路和平时的电子制作调试中性能还是能够满足要求的，不足 500 元的成本我觉得是很实用的，特别是对象我这样的穷学生电子爱好者。

谁对自制数字示波器有兴趣？

想体验一下兴奋到失眠的感觉？

跟我来吧，我们一同体验这种兴奋的感觉！

魏坤

2008 11 15

电路才是“硬”道理

——硬件电路简述

通过我的“蛊惑”，想必大家都想自制一台示波器玩玩，那就继续跟着我走吧！

所有的电子设备都离不开硬件，首先让我来对它的硬件结构进行一下简述：

总体电路如系统框图所示（图 1），前面已讲过，为了提高性能本电路采用“双核”结构，两片 AVR 单片机协同工作，MCU1 用于控制和频率测量，MCU2 用于数据处理和显示控制，两片单片机采用 SPI 总线通信。

信号从探头输入，进入程控放大（衰减）电路进行放大（衰减），再对被放大（衰减）的信号进行电平调整后送入高速 AD 转换器对信号进行采样，采样所得的数据存入 FIFO 存储器中，当 FIFO 存满后通知 MCU2，MCU2 从 FIFO 存储器中读出数据进行处理，将波形显示在 LCD 模块上。时钟电路为高速 AD 转换器和 FIFO 存储器提供从 600Hz~60MHz 的 8 种不同的频率信号作为不同水平扫速时的采样时钟频率。从程控放大器输出的信号一路送入 AD 转换器，另一路送入整形电路对输入信号进行整形，作为测频率的待测信号送入 MCU1 的 16 位计数器外部触发引脚 T1（PD5），进行频率测量，程控放大器的放大（衰减）倍数和时钟电路的输出频率均由 MCU1 控制。MCU1 将被测信号的频率、程控放大器的放大倍数和时钟电路的输出频率等数据通过 SPI 总线发送给 MCU2，MCU2 以这些数据作为频率、水平扫速、灵敏度和峰峰值计算、显示的依据。

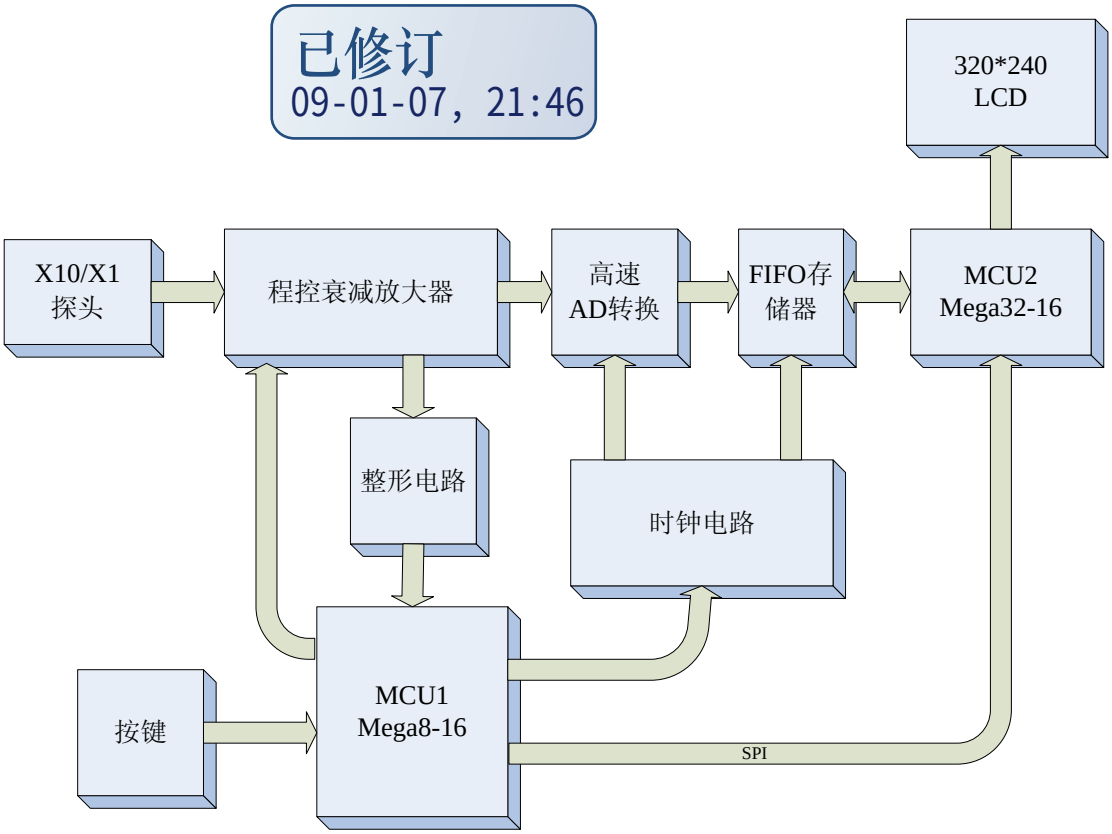


图 1：系统框图

下面就各个模块电路做以简单介绍。

1.程控放大电路和电源电路：

将程控放大电路与电源电路放在一块讲，是因为他们不仅有着密切的联系，而且还是做在一块电路板上的。

程控放大器的作用是对输入信号进行衰减或放大调整，使输出信号电压在 AD 转换器输入电压要求范围内，达到最好的测量与观察效果，所以程控放大器电路在规定带宽内的增益一定要平坦，故对运算放大器的要求比较高，在本电路中我选用的是 NSC 公司生产的高速运算放大器 LM6172 双运放，带宽为 100MHz，转换速率 3000v/μs,每通道消耗电流 2.3mA，输出电流可达 50mA，完全满足本电路的要求，选择该芯片的另一个原因是价格，邮购价格为 8 元一片，相比 ADI,MAX 等公司几十元一片的高速运放芯片来说算是很廉价了，电源采用正负双电源供电，由于整个电路总的电源输入为单 8v，所以专门用一片 dc/dc 电路 MC34063 为其构成了负压转换器再经稳压得到-5v 电压，+5v 通过对输入电压稳压得到。

程控放大器电路如图 1 所示，被测信号从 BNC 插孔输入，S0 继电器决定输入耦合方式，S0 吸合为直流偶和方式，S0 断开为交流耦合方式。信号通过交直流耦合选择开关后被送入由 R3~R5 和 C2~C4 组成的 X0.5/ X0.05 的衰减电路，衰减倍数由 S1 控制，当 S1 未吸合时

接在“0”端，对应的衰减被数为 $\frac{R_4+R_5}{R_3+R_4+R_5}=0.5$，当 S1 吸合时接在“1”端，对应的

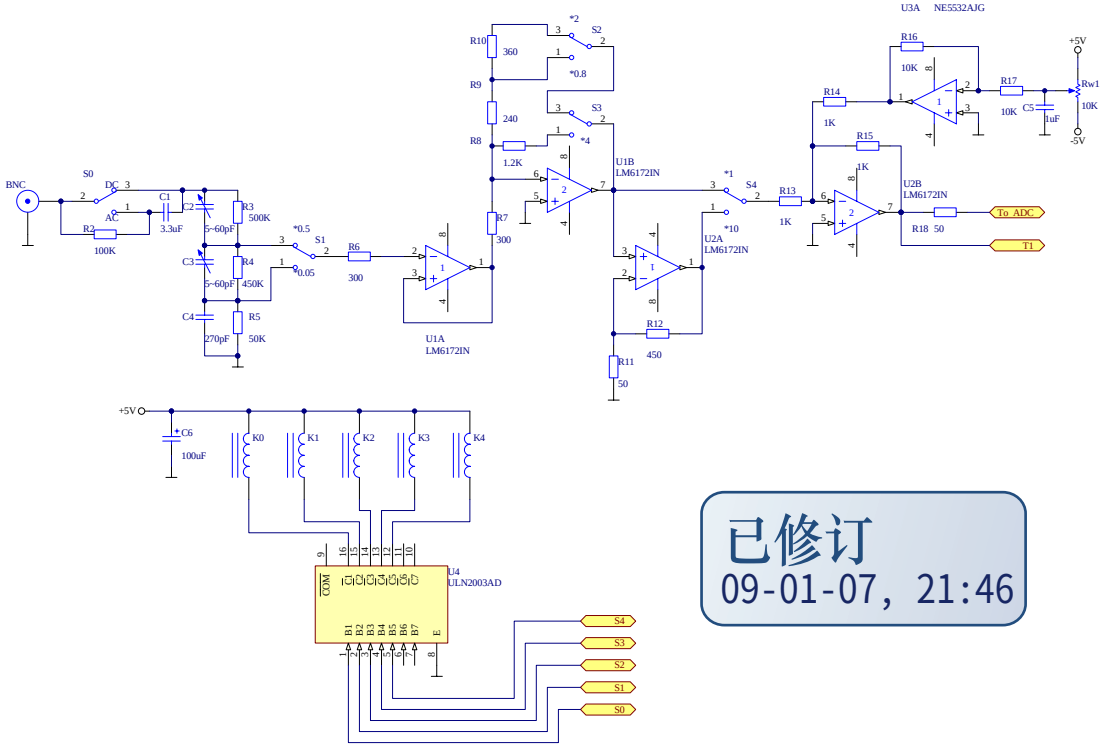


图 1：.程控放大器电路

衰减被数为 $\frac{R_5}{R_3+R_4+R_5}=0.05$，C2、C3 对高频信号进行补偿。经过衰减的信号进入由高速运算放大器 U1A 组成的缓冲器缓冲，然后被送入由 U1B 组成的 X-0.8/ X-2/ X-4 的反相放大电路，放大倍数由 S2 和 S3 控制，当 S2、S3 均未吸合时对应的放大倍数为 $-\frac{R_9+R_{10}}{R_7}=-2$，当 S2 吸合 S3 未吸合对应的放大倍数为 $-\frac{R_9}{R_7}=-0.8$，当 S3 吸合则不用考虑 S2 的情况，但为降低功耗使 S2 断开，此时对应的放大倍数为 $-\frac{R_8}{R_7}=-4$。输出信号又通过 S4 选择是否经由 U2A 组成的同相放大器放大，当 S4 未吸合，则不经过同相放大，当 S4 吸合，则信号被放大 $\frac{R_{11}+R_{12}}{R_{11}}=10$ 倍，最后信号被送入由 U2B 组成的放大倍数为-1 倍的反相放大器来消掉由第一级反相放大器所带来的负号，与此同时 U3A 送来的反相基线电压由 U2B 反相后作为 AD 转换器的输入中点电压被叠加在被测信号上被送入 AD 转换器，因为 ADS830E 的模拟

输入电压范围是 1.5V~3.5V，输入中点电压为 2.5V，所以基线电压应为 2.5V。调节可变电位器 Rw1 将调整基线电压的值，从而调整基线的位置。程控放大电路的放大倍数以及垂直电压灵敏度与 S1~S4 的关系见表 1：

放大倍数	灵敏度	S1	S2	S3	S4
0.04	5V	H	H	L	L
0.1	2V	H	L	L	L
0.2	1V	H	L	H	L
0.4	0.5V	L	H	L	L
1	0.2V	L	L	L	L
2	0.1V	L	L	H	L
4	50mV	L	H	L	H
10	20mV	L	L	L	H
20	10mV	L	L	H	H

表 1：程控放大电路的放大倍数

“L”代表继电器未吸合，“H”代表继电器吸合，确定继电器的常闭触点和常开触点很重要，因为继电器的吸合需要消耗一定电流，我选用的继电器型号为 TO2-5V，吸合电流为 15mA。在常用的 3 个灵敏度上（0.5V/div，0.2V/div，0.1V/div）最多只有一个继电器吸合，继电器的驱动由 ULN2003 担任。这种由运算放大器构成组合程控放大器的思想也可用于别的放大电路，平时多总结积累电路模型对提高电路设计能力非常有帮助，这个电路你记下了吗？

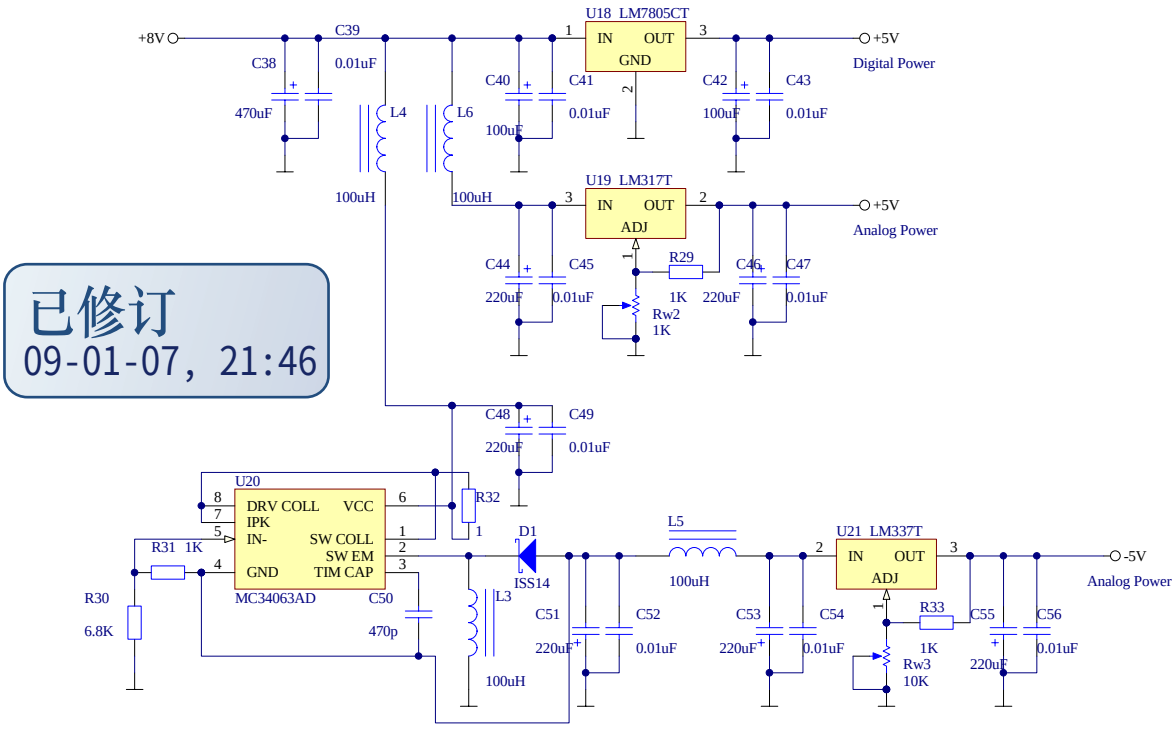


图 2：电源电路

电源电路为整个示波器提供能源，作用非常重要！电路见图 2 所示。

该示波器电路中供电分为数字和模拟两部分。为避免相互干扰，所以将数字部分的供电与模拟部分的供电分开，分别用独立的稳压电路，并用电感与电容做成的滤波器隔离。数字部分需要单+5V 电源，由一片 LM7805 对 8V 电源电压稳压得到。模拟部分主要是程控放大器电路和 AD 转换器的模拟输入电路，程控放大器电路需要±5V 双电源，AD 转换电路的模拟部分需要+5V 的单电源，+5V 电压由 LM317T 对 8V 电源电压稳压得到，而-5V 电压专门用一片 DC/DC 芯片 MC34063 将+8V 转换成约-8.3V，DC/DC 输出电压由 R30 和 R31 决定，$V_{OUT}=-\frac{R_{30}+R_{31}}{R_{31}}\times 1.25V=-8.25V$ $V_{OUT}=-\frac{R_{30}+R_{31}}{R_{31}}\times 1.25V=-8.25V$，输出的负电压由负压稳压芯片 LM337 稳压得到-5V，为避免 DC/DC 电路对其他电路产生干扰，在其输入和输出端分别串联 L4 和 L5 进行隔离，在选择元件时蓄能电感 L3 选择磁罐封装带屏蔽的电感，使干扰降到最低。

2. 高速 AD 转换与 FIFO 存储电路

数字示波器中最重要的电路是 AD 转换电路，它的作用是将被测信号采样并转换成数字信号存入存储器，说它是数字示波器的咽喉一点也不为过，因为它直接决定着数字示波器所能测量的最高频率，根据乃奎斯特定理，采样频率至少是被测信号最高频率的 2 倍才能复现出被测信号。而在数字示波器中采样频率至少应该是被测信号频率的 5~8 倍才行，否则根本观察不到信号的波形。在本电路中我选用的 AD 转换芯片为 BB 公司的 8 位高速 AD 转换器 ADS830E,官方资料给出的采样频率为 10kSa/s~60MSa/s, 通过实验发现转换速率在 1K 以下工作也很正常，所以本示波器的最低采样频率为 600Sa/s，要说明的一点是高速 AD 转换器一般都有高低端转换速率的限制，比如 TLC5540，8 位 AD 转换器，转换速率为 5MSa/s~40MSa/s，我试过当转换频率降到 2M 以下时就不能正常工作，所以选择 AD 转换芯片时不仅要注意最高转换速率还要关注最低转换速率，否则可能导致电路无法正常工作。有朋友也许会问 8 位转换精度会不会有点太低？其实 8 位转换器对于示波器来说是够用的，就拿这个电路来说，我选用的 LCD 显示模块的分辨率为 320*240，垂直分辨率为 240 格，而 8 为转换精度的分辨率为 256 格，比显示器的分辨率还高，所以绝对够用。还有就是价格及电路的设计，在最高采样率相同的情况下 10 位 AD 转换芯片的价格是 8 位 AD 转换芯片的几倍，而且位数的增加也使电路的复杂程度大大增加，将直接影响处理速度，导致屏幕刷新过慢，反而影响性能。所以本着够用的原则本示波器选用 60M 的 8 位 AD 转换芯片 ADS830E。

引脚排列见图 6。

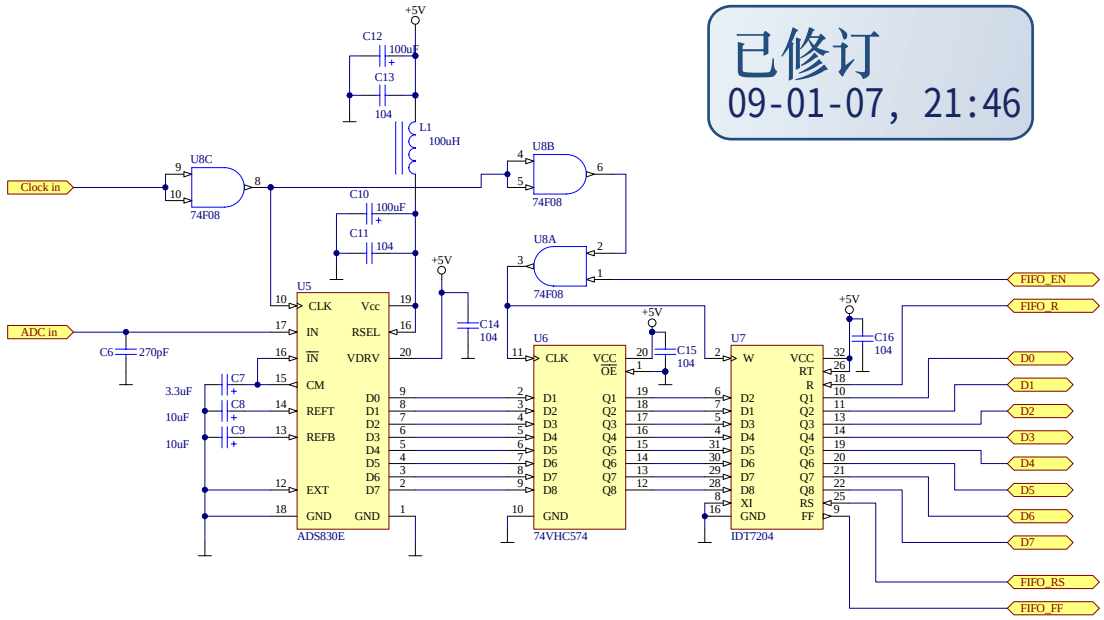


图 3：高速 AD 转换与 FIFO 存储电路

AD 转换与 FIFO 存储电路见原理图 3，由程控放大电路调整后的信号分成两路，一路进入 AD 转换电路进行采样，采样所得的数据由 74LVC574 锁存缓冲后送入 FIFO 存储器。FIFO 存储器是一种双口的 SRAM，（FIFO：frist in frist out，即先进先出存储器）这种存储器没有地址线，随着写入或读取信号对数据地址指针进行递加或递减，来实现寻址。在 AD 转换器与 MCU2 之间加入 FIFO 的作用是起到高速数据缓冲的作用，因为 AD 转换器的最高工作频率为 60MHz 远高于 MCU2 的工作频率，所以让 FIFO 与 AD 转换器同步工作存储 AD 转换器的转换输出数据。FIFO 存储器有 3 个标志位引脚，分别为 FF（满标志）：当存储器存满后置位该标志，此时存储器忽略一切写数据操作。HF（半满标志）：当存储器存满一半后置位该标志。EF（空标志）：当存储器被读空时置位该标志，此时存储器忽略一切读数据操作。FIFO 存储器结构图见图 4。本电路中只用了该芯片的 FF 标志与 MCU2 的 PB3 相接，当 FIFO 存储器存满后 FF 引脚被拉高，通知 MCU2 进行数据读取，这时 MCU2 禁止 AD 转换器与 FIFO 存储器的时钟，FIFO 的控制权交给 MCU2，（其实 MCU2 只是禁止了 FIFO 存储器的写时钟，见图 3，时钟信号通过 U8C 组成的缓冲器后直接加给了 ADS830E，所以 MCU2 不能禁止 AD 转换时钟，只能通过由与门 U8A 组成的时钟控制开关禁止或使能 FIFO 存储器的时钟信号。因为实验中发现 AD 转换在启动后的几个时钟周期内的采样不可靠，所以就让 AD 转换器一直工作，通过控制 FIFO 存储器来控制 AD 采样。在 AD 转换电路与 FIFO 存储器中加入 74LVC574 的目的是所存数据提高数据通道的稳定性。）当 MCU2 读完数据并完成软件触发后使能 AD 转换器与 FIFO 存储器时钟，继续读取新的数据，同时 MCU2 对读

取的数据进行处理、显示。

这儿再将高速数模转换器 ADS830E 的工作简单介绍一下，ADS830E 的时序如图 5 所示，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 2Vpp。当 11 脚置低电平时，输入电压范围是 2V~3V，即 1Vpp。进行程控放大器设计时要考虑这个问题，本电路选用 2Vpp 的输入电压范围。

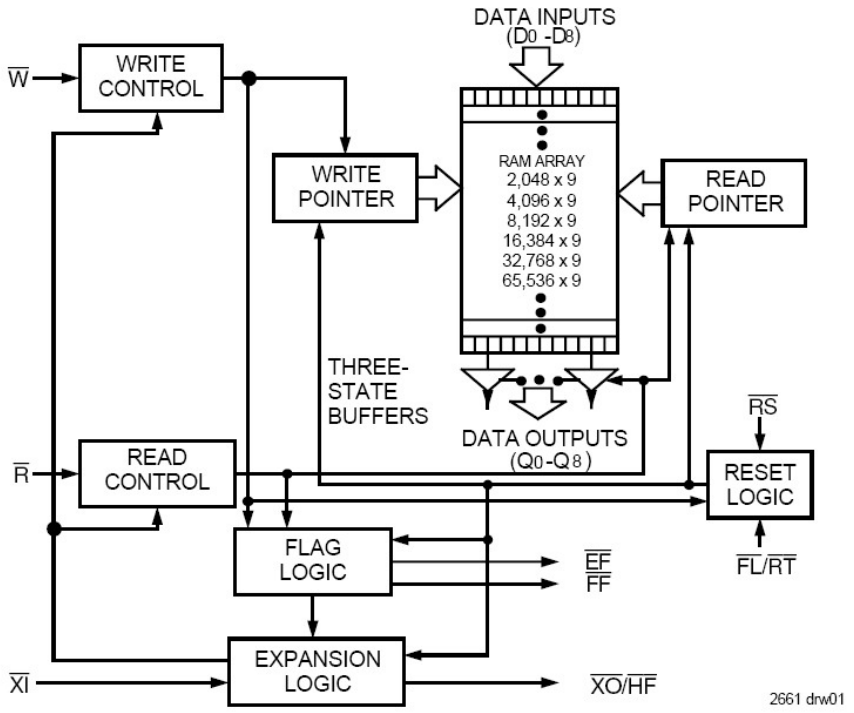
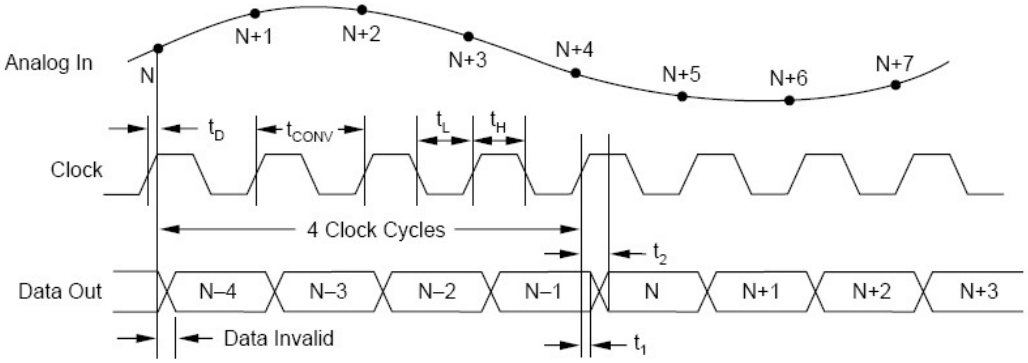


图 4：FIFO 存储器结构图



SYMBOL	DESCRIPTION	MIN	TYP	MAX	UNITS
t_{CONV}	Convert Clock Period	16.6		100µs	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

图 5：AD 转换时序图

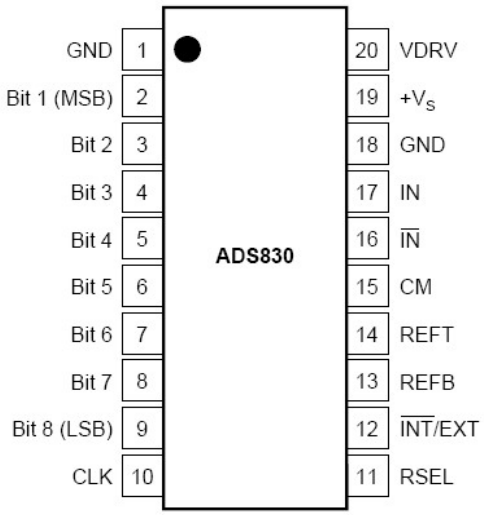


图 6：ADS830E 引脚图

3. 时钟产生电路

时钟产生电路为 AD 转换器提供一系列的采样时钟信号，分别为 600Hz、6kHz、60kHz、600kHz、3MHz、6MHz、30MHz 和 60MHz，共 8 种，分别对应着不同的水平水平扫速，由 MCU1 控制，控制关系见表 2。

时钟产生电路见图 7，基准时钟信号由一块 60MHz 的温度补偿型有源晶体模块提供，

输出的 60MHz 信号一路直接作为 60MHz 采样时钟送入多路选择器 74F151，另一路被送入由 74F74 触发器组成的 2 分频器分频，得到 30MHz 的信号分为两路，一路送入多路选择器 74F151，另一路送入由 2-5-10 分频器 74LS390 组成的 5 分频器进行分频，得到 6MHz 信号，再分为两路，一路继续分频，另一路送入多路选择器 74F151,后面几级分频与以上相同。对 60MHz 信号进行第一次二分频没有用 74LS390 中的 2 分频器，而单独使用了一片 74F74，是因为 74LS390 中的二分频器的最高输入频率为 40MHz，所以在其前面用了一级独立的二分频器。8 种时钟信号都被送入多路选择器，MUC1 通过对 74F151 的 S0、S1、S2 三根选通信号线进行控制来选择所需的采样频率。

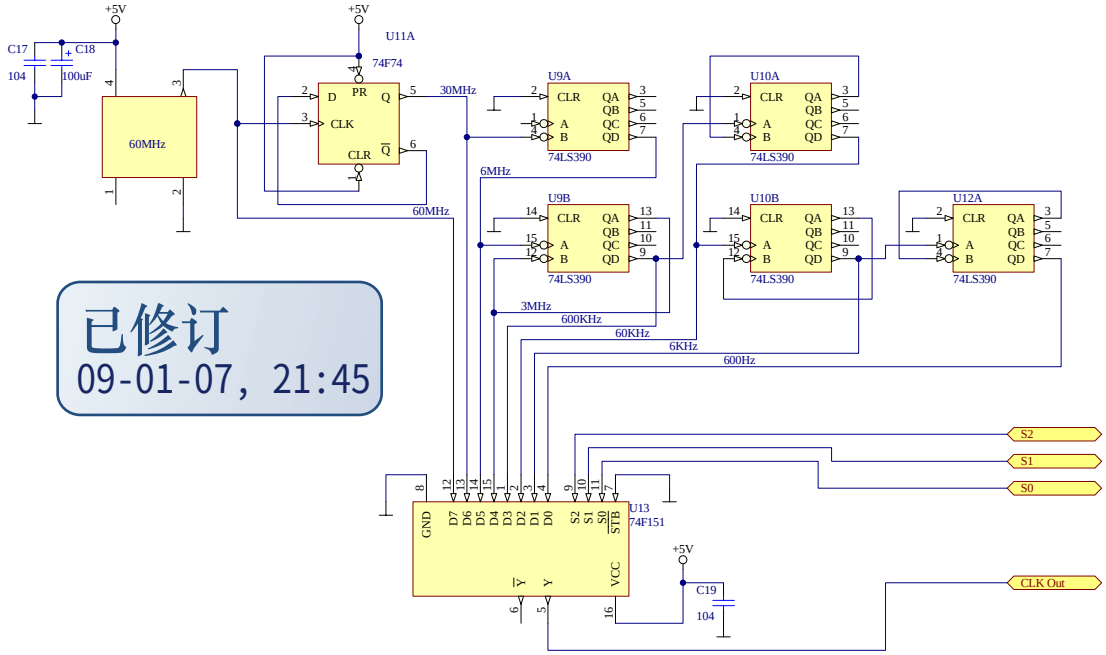


图 7：时钟产生电路

采样时钟频率	水平扫速	S0	S1	S2
600Hz	50ms	H	H	H
6kHz	5ms	L	H	H
60kHz	500us	H	L	H
600kHz	50us	L	L	H
3MHz	10us	H	H	L
6MHz	5us	L	H	L
30MHz	1us	H	L	L
60MHz	500ns	L	L	L
60MHz	250ns*	L	L	L

表 2：时钟频率、水平扫速和时钟控制数据关系表

4. MCU2 单片机显示处理电路

MCU2 选用 ATMEL 公司的 AVR 单片机 Mega32-16AI，与 51 单片机相比 AVR 单片机

具有更高的工作频率与更高效率的硬件结构，51 单片机的指令周期是将晶体振荡器的振荡频率进行 12 分频后得到的，又有累加器 Acc 在高速执行指令时的瓶颈因素，而 AVR 单片机则不同，它的指令周期就是晶体振荡器的振荡周期，有 32 个类似与累加器 Acc 的寄存器直接和运算器相连，取址周期短，又可预取指令实现流水作业，故可高速执行指令。Mega32-16AI 的 ROM 容量为 32KB，RAM 为 2KB，32 个输入输出口，官方给出的最高速度为 16MHz，但在实际使用中工作在 18~20MHz 也很稳定，所以用该单片机做显示处理非常合适。在本电路中为了提高 LCD 显示器的屏幕刷新速率所以使其工作在 18MHz，实际使用中电路工作十分正常。

MCU2 电路见图 8 所示。PD0~PD7 与 LCD 显示器 8 位并行数据端相连，PC1~PC5 与 LCD 显示器的控制端相连用于驱动 LCD 显示器（LCD 显示器资料见光盘），PC0 用于控制 LCD 背光，PC0=0 有背光，PC0=1 无背光。PB4、PB5 与 PB7 作为 SPI 通信端口与 MCU1 相连进行两个单片机之间的通信。PA0~PA7 与 FIFO 存储器的数据输出端 Q1~Q8 相连接，PB0~PB3 分别与 FIFO 存储器的使能（FIFO_EN）控制端、复位（FIFO_RES）控制端、读数据（FIFO_R）控制端和满标志（FIFO_FF）位相接。上电时，MCU2 通过 FIFO_RES 端口对 FIFO 存储器进行复位，复位后存储器的读写指针都指向 0，允许写数据，MCU2 通过 FIFO_EN 端使能 FIFO 存储器，开始将 AD 转换器输出的数据写入存储器，当 FIFO 存储器写满数据后 FIFO_FF 位被拉高通知 MCU2 读取采样数据，MCU2 禁止 FIFO 存储器写入数据，然后从 FIFO 存储器中读数据，当数据读完并完成软件触发后使能 FIFO 存储器继续存储采样数据，然后从读取的数据中测出波形的峰峰值后将数据转换成波型与参数显示在 LCD 显示器上，峰峰值的测量是通过对一屏显示数据进行比较取出最大值与最小值与当前垂直电压灵敏度作为系数计算出来的。SPI 通信通过中断的方式实现，MCU1 每次给 MCU2 发送频率、水平扫速、垂直电压灵敏度等数据一共为 9 个字节，每发送一个字节 MCU2 中断一次将接收到的数据存到一个数组中，直到 9 个字节全部发送完毕 MCU2 才对接收到的数据进行处理显示。这样可以使 MCU2 在平时都工作在数据的处理和显示上，提高了数据的处理速度。

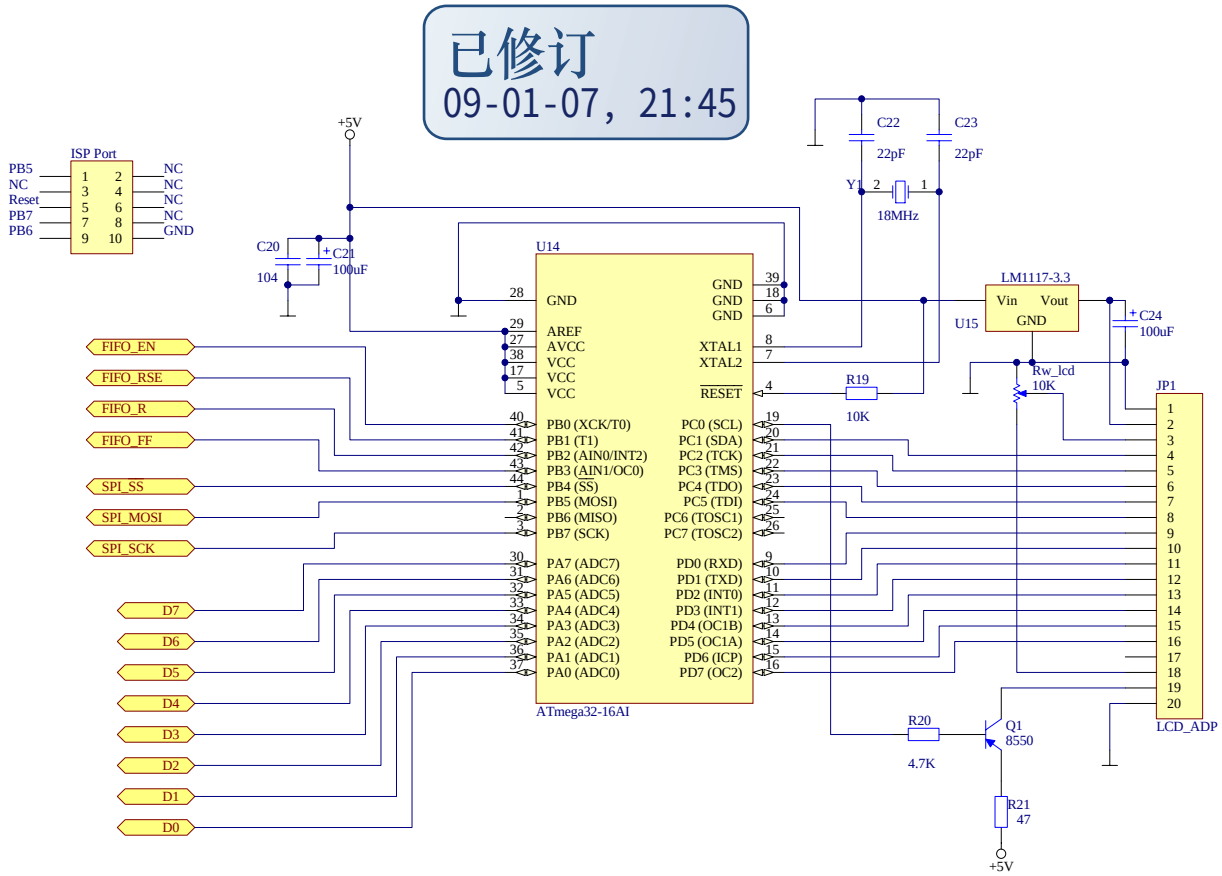


图 8：MCU2 控制电路图

5. MCU1 单片机控制与信号整形电路

MCU1 同 MCU2 一样也选用 AVR 单片机，型号为 Mega8-16，工作频率为 16MHz，在电路中负责控制程控放大器和时钟发生电路并负责测量被测信号频率，将各种参数通过 SPI 总线发送给 MCU2。

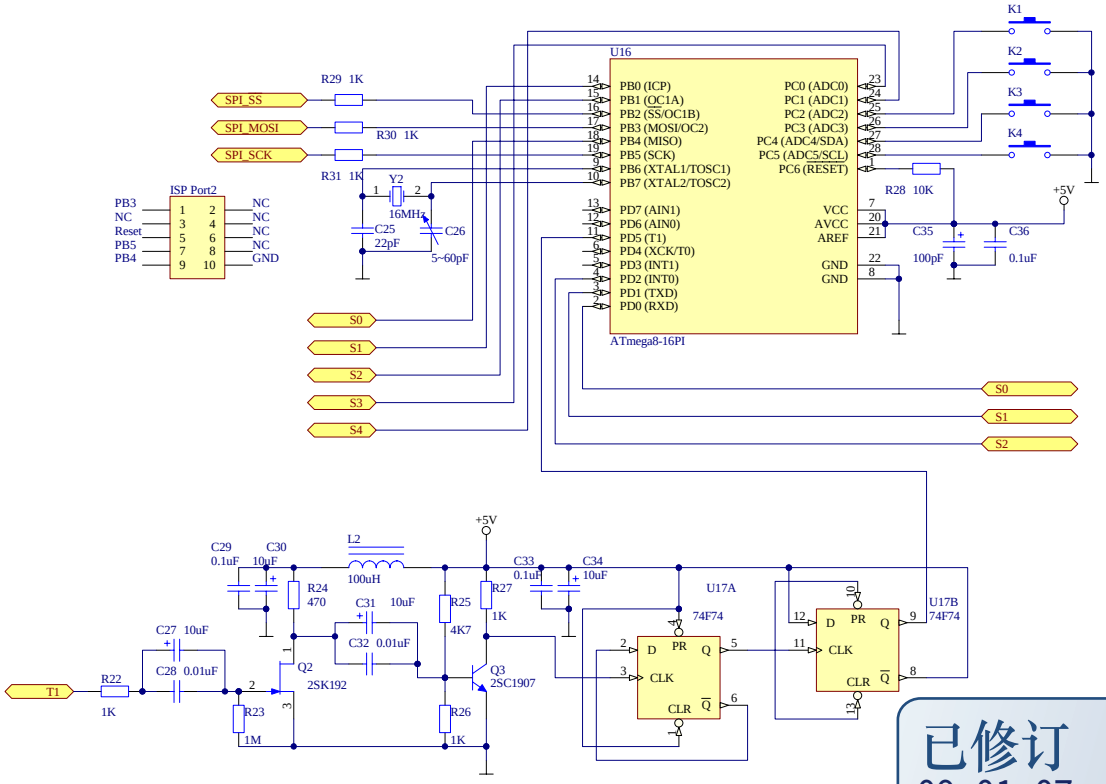


图 9：MCU1 及整形电路图

MCU1 电路见图 9， PC2~PC5 共 4 个 IO 口接 4 个轻触开关 S1~S4，S1、S2 是两个复用键，用于控制水平扫速与垂直电压灵敏度，功能通过 S4 切换，当前功能状态显示在显示器上，如果当前的控制功能为控制水平扫速则在显示器的右下边反显示“T”，如果当前的控制功能为控制垂直电压灵敏度则在显示器的右下边反显示“V”，见照片。S3 是触发控制，当前状态显示在控制状态左边，箭头上升则自动触发，箭头向下则不触发。长按 S3 选择交直流偶和方式。该示波器现在只能实现基本功能，其他更多功能有待于广大爱好者共同努力。S0、S1、S2、S3、S4 共 5 个端口分别连接 PB4、PB0、PB1、PC0、PC1 用于垂直电压灵敏度控制，控制数据见“程控放大电路”中的表 1。PB2、PB3、PB5 作为 SPI 总线接口与 MCU2 通信，为了防止下载程序时两芯片 SPI 口冲突所以在两单片机之间的 SPI 总线上串联 3 只 1K 的电阻，实验证明此法非常有效，电路工作稳定。PD0~PD2 共 3 个 IO 口用于时钟控制，控制数据见“时钟产生电路”中的表 2。频率的测量使用了 16 位计数器，外部下降沿出发。程控放大器输出的信号送给由场效应管 Q1 和高频三级管 Q2 组成的高输入阻抗整形电路整形后再由 U17 触发器 74F74 进行 4 分频然后送入 MCU1 的 T1（PD5）脚进行计数测频，在低水平扫速时（5ms/div 和 50ms/div）为了保证测频精度侧频周期为 1s，在高水平扫速时（小于 5ms/div）测频周期为 0.25s。测频的原理是通过记录 1s 或 0.25s 内计数器记录的脉冲数来换算频率，测频定时由中断完成，每测完一次频率就向 MCU2 通过 SPI 总线发送一次数据，

所以在高水平扫速时每秒向 MCU2 发送 4 次数据，而在低水平扫速时每秒向 MCU2 发送 1 次数据，能较好的保证参数显示的实时性。

魏坤

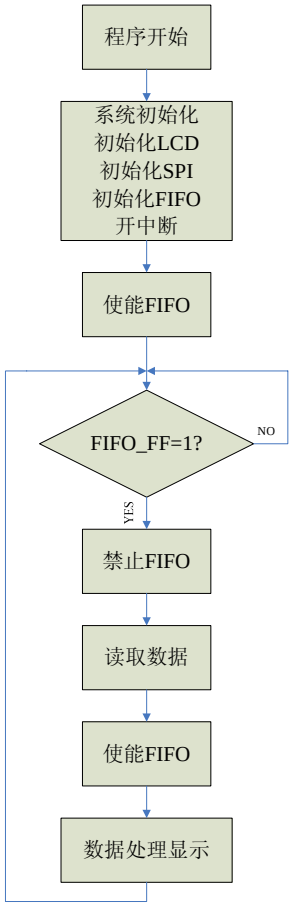
2008 11 15

给你的电路注入灵魂

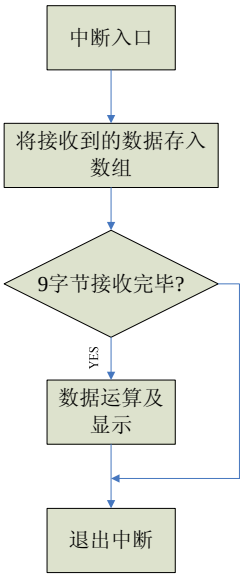
——程序设计

没有程序那一堆堆电路起不到任何作用，完全是一堆废板子！就像一台没有操作系统的电脑一样，只能废电。程序设计是整个示波器制作中的难点，本文将详细讲解程序的设计。

该示波器中的程序全部是用 c 语言编写的，开发环境为 CodeVisionAVR C，原程序在附件中，下面就各个重要的子程序的设计一一叙述，其它程序见原程序。MCU2 与 MCU1 的程序流程图分别见图 1 和图 2。



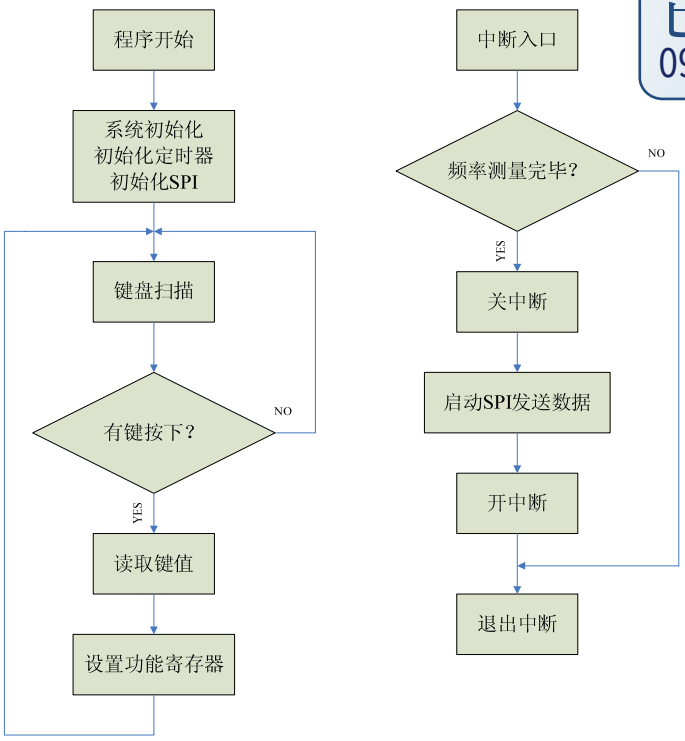
MCU2主程序流程图



MCU2中断流程图

已修订
09-01-07, 21:49

图 1：MCU2 程序流程图



MCU1主程序流程图　　MCU1中断流程图

图 2：MCU1 程序流程图

1．同步触发的软实现

细心的朋友会发现这个示波器电路中缺少一部分电路，就是硬件触发电路，为了降低电路的复杂性我在做这个示波器时没有做这个电路，而是用软件实现同步触发的，这样做有个弊端就是几乎无法实现单次触发，因为我基本不用这个功能，需用这个功能的朋友只需在程控放大器部分加上一个由高速比较器构成的迟滞比较器然后将输出端接到一个外部中断的输入口即可。当然程序和电路就要做相应的变化，这里就不多讲了。软件触发的好处是触发条件更易调整，只需调整比较语句中的参数即可。保证可以用软件触发的条件是要有足够大的存储空间，显示一屏的数据为 240 个，但每次读进单片机的数据为 500 个，多余 260 个数据就是作为不满足触发条件的舍弃余量，为了以防万一，当从 500 个数据中以经读出超过 260 个数据但还没有符合触发条件的数据时，将跳出触发比较循环，重新从 FIFO 存储器中读出 500 个数据，因为 FIFO 存储器为 4K 容量，最多可以这样重复读取 8 次数据，所以软触发可以非常稳定的工作，在该示波器的 MCU2 中控制触发的语句见以下程序段：

```
read:
for(i=0;i<500;i++)     //从 FIFO 存储器中读 500 个数据
{
```

```
  FIFO_R=0;
  add[i]=FIFO_bus;
  FIFO_R=1;
 }
while(!(add[q]<=m&&add[q+1]>=m))     //满足幅度为 m 且为上升沿则触发
   {
   q+=1;
   if(q>=260)          //若存储数据不足则重新读数据
    goto read;
   }
```

程序的意思是只有当此时采样信号的数值是 m 且为上升沿时才可以触发，改变触发沿只需改变运算符，改变触发电压只需改变 m 的值即可，m 的取值范围是 0~255。

2. 从采样数据中测信号峰峰值

本示波器就能够测量输入信号电压的峰峰值，并显示在屏幕上。这个功能由峰峰值测量子程序完成，见下面的程序段。

在程序开始时给a中赋值128，即基线电压值。因为一屏幕的显示数据为240个，所以用for（）循环将if…else…判断语句执行240次，在a中存放最大值，在b中存放最小值。对每个数据进行比较，如果该数据比a大则将这个数据存入a，如果小于a，则将这个数据与b进行比较，比b大则抛弃，比b小则存入b。故当240个循环执行完后a中存放的是这一屏幕数据中的最大值，b中存放的是这一屏幕显示数据中的最小值。在比较完后用a减去b，得到差值存入c中，则c中保存的值就是电压的峰峰值，调用电压计算显示子程序根据当前的垂直灵敏度给c乘以不同的倍数，得到实际的峰峰值。当前垂直灵敏度的判断由一个switch（）选择结构完成。biao寄存器中的数据是当前的垂直灵敏度，case 4：后面没有运算是因为程控放大器在此状态下的放大倍数为1，即没有放大也没有衰减。

在计算完峰峰值后，设置LCD显示器，使其工作在文本模式（因为只有在文本模式下对字库的调用才有效），然后设置屏幕上显示电压峰峰的坐标（对该LCD模块的控制是先送命令，后送参数。例如设置X坐标“SdCmd(0x60);SdCmd(30);”中，第一个SdCmd（）送的0x60是设置X坐标的命令，第二个SdCmd（）送的30是X轴的坐标，其他设置相同。具体见光盘中LCD显示屏的资料。），在设置完LCD后约定显示格式，小数点后保留2为有效数字，显示单位为Vpp，显示完毕后需重新设置LCD工作状态使其工作在图形模式用于波形显示。

```
                    a=128;
                    for(i=0;i<240;i++)       //取数据中的最大值与最小值
                     {
                         if(add[i]>a)
                              {
                                a=add[i];
                              }
                          else if(add[i]<b)
                                {
                                  b=add[i];
                                }
                     }          //取得最大值存于a中，最小值存于b中
    c=a-b;        //取差值存入c中
    if(e>5)       //避免频繁换数据
          {
        disp_volt();   //调用电压值计算显示子程序
        e=0;
         }
         e++;
void disp_volt()   //电压值计算显示子程序
{
         c=c*0.667;
         switch(biao)//根据不同的垂直灵敏度计算峰峰值
                    {
                    case 0:c=c*25;break;
                    case 1:c=c*10;break;
                    case 2:c=c*5;break;
                    case 3:c=c*2.5;break;
                    case 4:;break;
                    case 5:c=c*0.5;break;
```

已修订
09-01-07, 21:49

```
                    case 6:c=c*0.25;break;
                    case 7:c=c*0.1;break;
                    case 8:c=c*0.05;break;
                    default:break;
                    }
SdCmd(0x00);SdCmd(0xcd); //设置WLCR 寄存器，使LCD工作与显示文本状态
SdCmd(0xf1);SdCmd(0x1f); //字型水平、垂直方向各放大2 倍
SdCmd(0x60);SdCmd(30);  //设置显示X坐标
SdCmd(0x70);SdCmd(50);  //设置显示Y坐标
sprintf(lcd_buffer,"%3d.%02dVpp",c/100,c%100); //约定显示格式，小数点后保留两位
ShowText(lcd_buffer);                          //有效数字
SdCmd(0x00);SdCmd(0xc5); //设置WLCR 寄存器,使LCD工作于图形显示模式
}
```

3．将采样数据转换成显示数据

LCD显示屏为320×240点阵的图形显示模块，内置RA8803 控制器。模块不仅可以显示单一的文本、图形，而且可以实现双图层的（“或”、“异或”、“同或”、“与”四种逻辑关系）合成显示。在本示波器中显示格线与波形是在不同的层上显示，显示关系为“或”，画方格线的程序见原程序，比较简单就不多说了，着重解释一下如何将采样数据转换成显示数据。

显示屏的地址结构见图3，由图可知对显示数据的操作最小单位为字节，因为Mega32的内存为2K字节，显示波形的区域为240*240，显示一屏波形所需处理的数据为7.2K,故Mega32不可能同时处理一屏波形的全部数据，所以将一屏波形按字节分为30列，每次处理一列，处理完后直接显示，然后处理下一列。将AD转换所得的数据作为给LCD显示器写数据的列地址，因为一列数据位240字节，所以定义一个容量为240字节的数组lcd_buffer[240]，lcd_buffer[]在初始时数据全为00H，因为每次对数据的操作至少是一个字节，而每次处理数据处理的是所显示一个点，所以对每列数据处理8次，定义一个变量m，在一列数据处理之前将其赋值为m=10000000B,处理该列第1个点时让该点垂直地址所对应的数组中的数据（00H）与m相或并将结果存入数组，再将变量m右移一位，即m=01000000B。让第2点垂直地址所对应的数组中的数据与m相或并将结果存入数组，再将变量m右移一位，即m=00100000B ……，这样直到一列数据中的8个点全处理完，重新给m赋值为m=10000000B,

然后送显示。为了有较好显示效果，将显示相邻的点用线连接起来，在处理第一个点时预读出第二个点的垂直坐标，与第一个点的垂直坐标进行比较，如果比第一个点的垂直坐标小则从第一个点向第二个点拉线，如果比第一个点的垂直坐标大则从第二个点向第一个点拉线。具体程序如下所示：

已修订
09-01-07, 21:48

```
for(j=0;j<30;j++)          //将一屏数据分为30列
    {
      m=0b10000000;         //
      for(i=j*8;i<(j+1)*8;i++)  //处理每列中的8个点
       {
        k=add[i]; // 读出采样数据作为垂直坐标
        lcd_buffer[k]=(lcd_buffer[k]|m); //让该坐标对应数据与m相或并原位保存
        if(add[i+q]<add[i+q+1])           //判断拉线方向
          {
           for(k=add[i+q];k<add[i+q+1];k++)
            {
            lcd_buffer[k]=(lcd_buffer[k]|m);
            }
          }
        else
          {
           for(k=add[i+q];k>add[i+q+1];k--)
            {
            lcd_buffer[k]=(lcd_buffer[k]|m);
            }
          }
           m>>=1;        //将m的值右移一位
       }

      for(h=0;h<240;h++)     //送显示
       {
          SdCmd(0x60);SdCmd(j);          //设置显示X坐标
          SdCmd(0x70);SdCmd(h);          //设置显示Y坐标
          SdData(lcd_buffer[h]);         //传送显示数据
          lcd_buffer[h]=0;               //将已送出数据的存储器单元清零
       }
   }
```

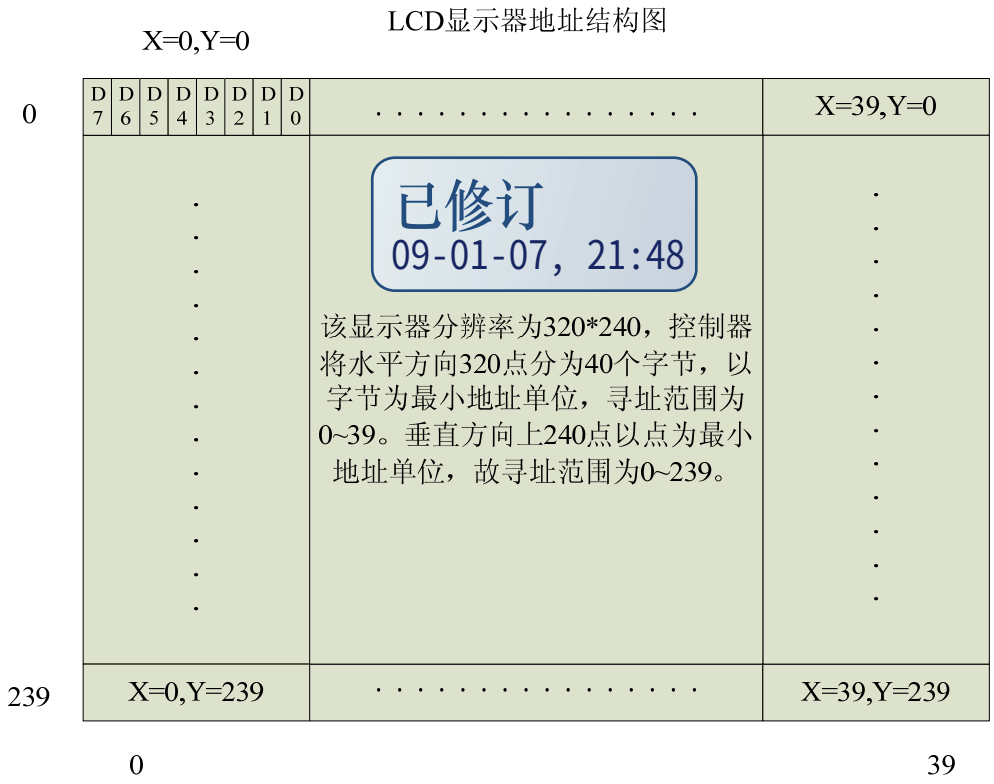


图 3：LCD 显示器地址结构图

4．用MCU1频率测量

用Mega8测量频率使用了其中的两个计数器/定时器。设置TCCR1B=6使16位计数器/定时器T/C1工作在计数器方式，对外部T1（PD5）引脚输入的脉冲信号进行计数（下降沿触发）。设置TCCR2=15使T/C2工作在CTC模式，内部时钟1024分频（16M/1024=15.625KHz），设置OCR2=124使中断时间为(124+1)/15.625=8ms，在低水平扫速时每隔8ms中断一次，在高水平扫速时通过重新设置TCCR2=14则每隔2ms中断一次，在这里以低水平扫速时为例，每次T/C2的中断中都首先记录下T/C1寄存器TCNT1当前的计数值，因此前后两次寄存器TCNT1的差值（time1_new-time1_old）或（65536-time1_old+time1_new）就是8ms时间内T1引脚输入的脉冲个数，为了提高测量精度程序对125个8ms内的脉冲个数进行累计，将累计值存入变量freq中，即可知限定时间为1s内有多少个脉冲，这样就将T1脚上的脉冲频率测量出来了，而T1脚上的频率是经过4分频后的所以真正的频率是将测量的频率的4倍。具体程序如下所示：

```
interrupt [TIM2_COMP] void timer2_comp_isr(void)
{
    time1_new = TCNT1;          // 8ms到，记录当前T/C1的计数值
    time_8ms_ok = 1;
```

已修订
09-01-07, 21:48

```
}
void time()
{
 if (time_8ms_ok)
        {   // 累计T/C1的计数值
            if (time1_new >= time1_old) freq = freq + (time1_new – time1_old);
            else freq = freq + (65536 – time1_old + time1_new);
            time1_old = time1_new;
            if (++freq_time >=125)
            {
                freq_time = 0;  // 1s到，
                freq_to_spibuff();// 将1s内的脉冲数送计算并通过SPI发送
                freq = 0;
            }
                time_8ms_ok = 0;
        }
}
```

5．将两个单片机联系起来

将两个单片机联系起来就是实现两个单片机之间的通信，在这里实际就是让 MCU1 控制 MCU2,为了完成这一功能所以使用 SPI 通信。

首先介绍一下 SPI 的通信协议：

SPI（串行外设接口）总线系统是一种同步串行外设接口，允许 MCU 与各种外围设备以串行方式进行通信、数据交换，广泛应用于各种工业控制领域。基于此标准，SPI 系统可以直接于各个厂家生产的多种标准外围器件直接接口。SPI 接口通常包含有 4 根线：串行时钟（SCK）、主机输入/从机输出数据线（MISO）、主机输出/从机输入数据线（MOSI）和低电平有效的从机选择线 SS。在从机选择线 SS 使能的前提下，主机的 SCK 脉冲将在数据线上传输主/从机的串行数据。主/从机的典型连接图如图 4 所示：

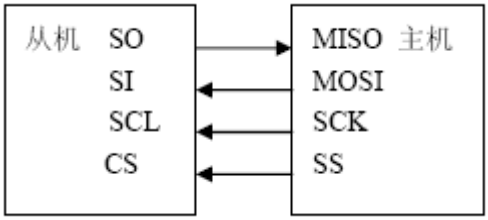


图 4：主/从机的连接图

串行外设接口 SPI 允许 AVR 单片机和外设之间进行高速的同步数据传输。AVR 单片机 SPI 的特点如下：全双工，3 线同步数据传输，主/从机操作，LSB 首先发送或 MSB 首先发送，7 种可编程的比特率，传送中断结束，写碰撞标志检测，可以从闲置模式唤醒。SPI 主机-从机的互连如图 5 所示，系统包括两个移位寄存器和一个主时钟发生器。通过将需要的从机的 SS 引脚拉低，主机启动一次通信过程。主机和从机将需要的数据放到相应的移位寄存器，主机在 SCK 引脚上产生时钟脉冲以交换数据。主机的数据从 MOSI 移出，从从机 MISO 移入。从机的数据从 MISO 移出，从从机 MOSI 移入。主机通过将从机的 SS 拉高实现与从机的同步。

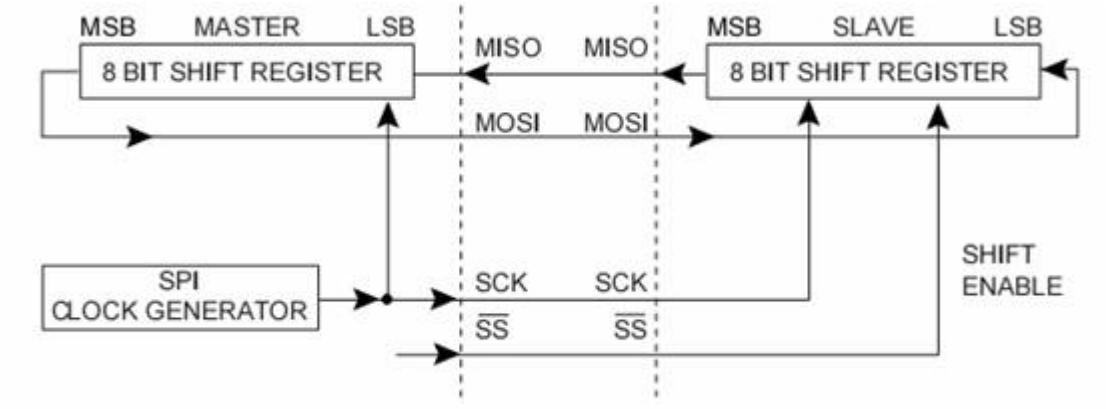


图 5：SPI 主机-从机的互连

下面将介绍 SPI 的几个特殊寄存器：

（1）SPI 的控制寄存器—SPCR

Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
读 / 写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

SPIE 为 SPI 中断使能，置位后，只要 SPSR 寄存器的 SPIF 和 SREG 寄存器的全局中断使能位置位，就会引发 SPI 中断。SPE 置位将使能 SPI，DORD 置位时数据的 LSB 首先发送；否则数据的 MSB 首先发送。MSTR 置位时选择主机模式，否则为从机。CPOL 置位表示空闲 SCK 为高电平；否则空闲时 SCK 为低电平。CPHA 决定数据是在 SCK 的起始沿采样还是在 SCK 的结束沿采样。通过对 SPR1、SPR0 进行设计，确定主机的 SCK 速率。

（2）SPI 的状态寄存器—SPSR

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	-	-	-	-	-	SPI2X	SPSR
读 / 写	R	R	R	R	R	R	R	R/W	
初始值	0	0	0	0	0	0	0	0	

SPIF 为中断标志位，串行发送结束后，SPIF 置位。若此时寄存器 SPCR 的 SPIE 和全局中断使能位置位，SPI 中断即产生。进入中断例程后 SPIF 将自动清零。在发送当中对 SPI 数据寄存器 SPDR 写数据将置位 WCOL，SPI2X 置位后 SPI 的速度加倍。

（3） SPI 的数据寄存器—SPDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	SPDR
读 / 写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	X	X	X	X	X	X	X	X	Undefined

SPDR 数据寄存器为读/写寄存器，用来在寄存器文件 SPI 移位寄存器之间传输数据。写寄存器将启动数据传输，读寄存器将读取寄存器的接收缓冲器。SPI 系统的发送方向只有一个缓冲器，而在接收方向有两个缓冲器。也就是说，在发送时一定要等到移位过程全部结束后才能对 SPI 数据寄存器执行写操作。而在接收数据时，需要在下一个字符移位过程结束之前通过访问 SPI 数据寄存器读取当前接收到的字符。否则第一个字节将丢失。

本示波器中只用 MCU1 控制 MCU2，所以 MCU1 只用于发送控制数据，而 MCU2 只用于接收控制数据，所以将 MCU1 配制成 SPI 主机，将 MCU2 配制成 SPI 从机即可。在实际的程序设计中由于 MCU1 启动 SPI 通信是在中断服务程序中完成，所以在执行完后相应寄存器会被清零，导致数据错误，所以 MCU1 并没有使用其中的 SPI 控制器，而是使用一个子程序模拟 SPI 通信，解决了控制寄存器被清零的问题。MCU2 则使用了本身的 SPI 控制器进行数据接收。具体程序见以下程序段：

（1）MCU1 模拟 SPI 主机程序段：

spi_out（）为 SPI 发送子程序，带有参数 j，即 j 为要发送的数据，发送数据时先拉低 ss，让从机开始接收数据，然后用 for（）循环将数据按由左至右的顺序（即高位先发送）发送给从机，具体方法是将 j 与 0b10000000 相与，屏蔽低 7 位，是 1 则将 dat 拉高，否则置低，然后拉高 clk，延时 lus 再置低 clk，模拟时钟信号，再将 j 左移一位，再与 0b10000000 相与，然后判断发送……直到 8 位数据发送完毕，拉高 ss 告诉从机数据发送完毕进行数据存储。发送数据时约定数据格式，即两个单片机之间的通信协议：每次发送 9 个字节，前 4 个字节是测得的频率数据，且高位在前；第 5 个字节为垂置灵敏度数据；第 6 个字节为触发控制数据；第 7 个字节为同步控制数据；第 8 个字节为水平扫速数据；第 9 个字节为功能复用键的当前功能标志。从机再接收到数据后按照这样的顺序对数据进行处理，实现相应的功能。spi_out()这个子程序还可以用于其他需要 SPI 控制的芯片，只需在调用前对 IO 口进行定义即可。

已修订
09-01-07, 21:48

```
spi_out(unsigned char j)
{
  unsigned char u;
  ss=0;
  for(u=0;u<8;u++)
       {
         if(j&0b10000000) {  dat=1; }
         else {dat=0;}
         delay_us(1);
         clk=1;
         delay_us(1);
         clk=0;
         delay_us(1);
         j<<=1;
       }
```

```
        delay_us(1);
    ss=1;
}
void freq_to_disbuff()
{            if(fr==0)
             {
             freq=freq*4;
             }
              eep=freq>>24;//取频率高 8 位
              spi_out(eep);
              delay_us(10);
              eep=(freq>>16)&0xff;
              spi_out(eep);
              delay_us(10);
              eep=(freq>>8)&0xff;
              spi_out(eep);
              delay_us(10);
              eep=freq&0xff;//取频率低 8 位
              spi_out(eep);
              delay_us(10);
              spi_out(w[i]);  //垂直灵敏度数据
              delay_us(10);
              spi_out(tri);   //触发数据
              delay_us(10);
              spi_out(hold);  //同步数据
              delay_us(10);
              spi_out(kr);   //扫速数据
```

```
delay_us(10);
spi_out(zhi);  //复用键功能标志数据
delay_us(10);
}
```

（2）MCU2 从机 SPI 程序段：

init_spi()函数是将 MCU2 配制成 SPI 从机，每接收一个字节的数据中断一次，中断服务程序中将接收到的数据存入数组，并将数组地址加 1，然后判断 9 个字节是否接收完毕，若没接收完则继续等待接收，接收完后则将数据按约定格式处理显示。大家可以根据自己的需求改变这些格式为其增加新的功能。

```
void  init_spi()     //SPI 初始化子函数
{
DDRB.7=0;
PORTB.7=1;
DDRB.5=0;
PORTB.5=1;
DDRB.4=0;
PORTB.4=1;              //将 SPI 端口设置成输入
SPCR=0b11000101;   //设置 SPI 为从机
SPSR=0X00;             //清零 SPSR 寄存器
}
interrupt[SPI_STC]  void  spi_isr(void)     //SPI 接收数据中断
{
 if(j!=9)
 {
     i[j]=SPDR;        //将接收到的数据存入数组
     j++;                 //给数组地址加 1 }
```

```
}
        if(j>=9)            //判断 9 个字节数据是否接受完毕
    {   eep=i[0];
    freq=eep;
    freq=freq<<8;
    eep=i[1];
    freq=freq|eep;
    freq=freq<<8;
    eep=i[2];
    freq=freq|eep;
    freq=freq<<8;
    eep=i[3];
    freq=freq|eep;       //将频率值整和到 freq 寄存器
    disp_freq();         //显示频率值
    eep=i[4];
    disp_volt();         //显示垂直灵敏度
    tri=i[5];            //显示触发方式
    hold=i[6];           //显示同步
    biao=i[7];           //显示扫速
    disp_time();
    zhi=i[8];        //显示复用键当前功能
    disp_cond();
    j=0;}
```

已修订
09-01-07, 21:48

魏坤

2008 11 15

神形合一，让你的示波器动起来！

——制作调试篇

硬件软件都讲完了，下来就将它们组装起来吧！

一：元件选择

制作本示波器所需的主要元件（包括显示器、集成电路和继电器）的型号、数量以及参考价格见表1，其中参考价格为我买该元件时的价格，仅供参考。加上电阻电容总的费用不超过500元，如果你对显示的要求不是很高，可以选择分辨率低一些的显示器那样成本会有较大幅度的下降，比如一块240*160的显示器价格为190元，而192*128的仅为140元，建议显示器的分辨率不要低于192*128。当然显示器更换了，驱动程序也要根据显示器的相关指令做相应的调整，这里就不多说了。其他的元件（电阻电容等）按图取值即可，程控放大器部分的一些特殊阻值的电阻可以通过电阻的串并联以及使用多圈电位器调整得到。该示波器的探头使用成品4MHz/40MHz探头。

器件型号	数量	器件功能描述	单价(元)
LM2068R	1	320*240 液晶显示屏	280
Mega32-16	1	AVR 单片机	17
Mega8-16	1	AVR 单片机	7
IDT7204-12	1	4KB、FIFO 存储器	40
ADS830E	1	8 位 60MSa/s AD 转换器	27
LM6172	2	高速运算放大器	8
NE5532	1	运算放大器	2
74F74	2	双触发器	1
74F08	1	高速 4-2 输入与门	1
74LS390	3	双 2-5-10 进制计数器	1
74F151	1	8 输入多路选择器	2
74LVC574	1	高速 8 位锁存器	1
ULN2003	1	7 路驱动器	2
MC34063	1	DC/DC 转换芯片	1
LM317	1	正电压三端稳压器	1.5

LM337	1	负电压三端稳压器	1.5
LM7805	2	+5V 三端稳压器	1
10cm*20cm 双面板	2		13
AQ-5V	5	微型继电器	3

表1：重要器件表

二．印刷板的设计及注意事项

印刷电路板使用热转印法制做，具体方法《无线电》上有详细介绍这里就不多讲了。

本制作中的一些电路板是用双面板做的，包括程控放大电路板、AD转换及FIFO存储器电路板和时钟电路板。在业余条件下制作双面都走线的电路板是比较困难的，首先是不容易将两面走线对准，其次过孔无法沉铜。所以我只做了双面覆铜板一面，另一面的铜箔全部用来做接地，方法是在需要接地的地方直接放上焊盘通过过孔接在背面的铜箔上，这样做的好处是起到电磁屏蔽作用而且布线相对单面板来说也更加简单，不用来接地的孔用大钻头进行轻微的扩孔削去周围的铜箔就行了，本电路中的程控放大电路、时钟电路和AD转换与FIFO存储器电路都是用这种办法做的，效果不错。

1.程控放大电路印刷板见图1，具体制作成电路板见图2和图3，从图1中可以看出电源电路的大部分都是与程控放大器做在同一块电路板上，印刷板上边为程控放大器电路部分，下边为负压产生和正负电源稳压电路，在安装电路时首先安装并调试电源电路，在电源电路装调好后再安装放大电路，因为电源电路装好后需要调整，如果先安装放大电路那么调整电源电路时就要断开很多线，很麻烦，所以先装调电源再装放大电路，可以省去很多重复工作，具体做法电路调试中有详细介绍。从图2中可以看到5个深蓝色的长方体块，这是进行耦合方式选择和放大（衰减）倍数控制的继电器，体积仅比一个555大一点，当然在选择继电器时还可以选择别的继电器，只是电路板要进行一些变化，我选择的这个继电器（AQ-5V）在使用中有一些问题需要注意，一般的继电器线圈绕组供电没有正负极之分，而该继电器有正负极之分，接错了将没有动作，其第1脚为电源正，第10脚为电源负，在设计电路时要特别注意。从图2中还可以看出正面的覆铜层全部用来接地，做法如前所述。

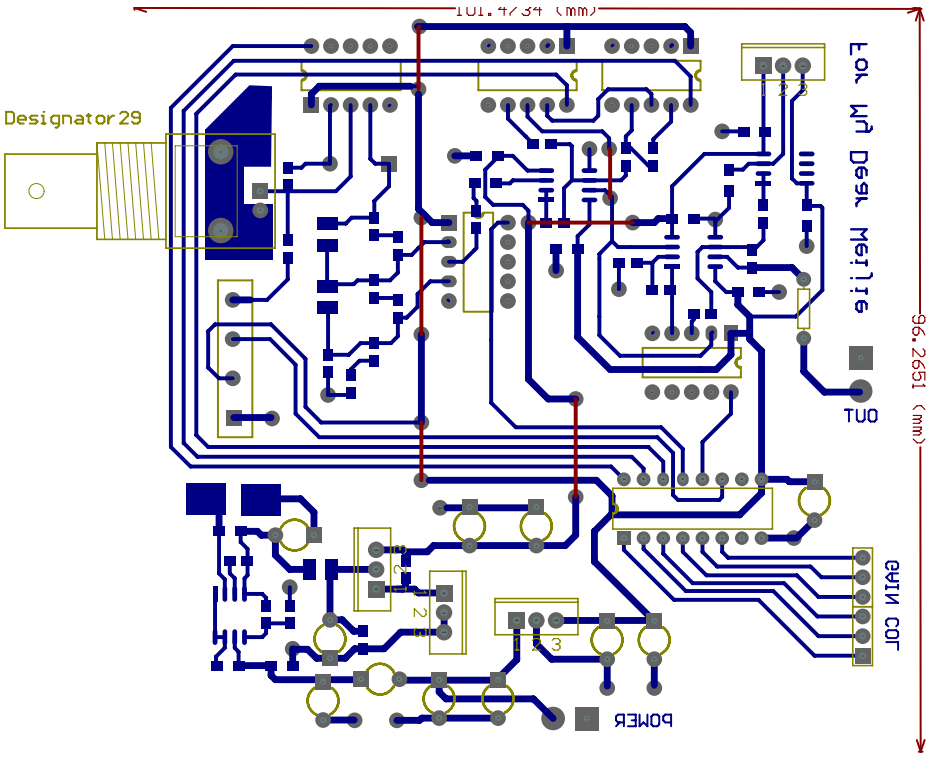


图 1：.程控放大器电路 PCB 板图

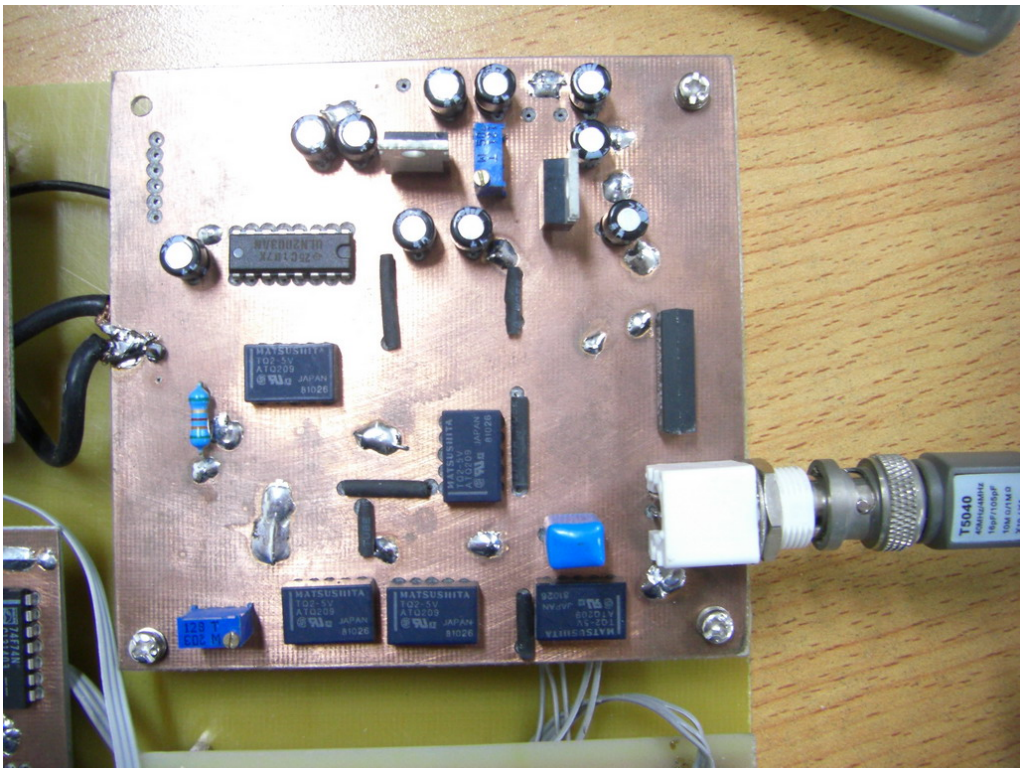


图 2：.程控放大器电路照片正面

图3为程控放大电路印刷板的背面，板子右上边有两个橘红色的可变电容，这两个电容的作用是对衰减后的信号进行边沿补偿，具体的调整后面会有详细介绍。

已修订
09-01-07 21:58

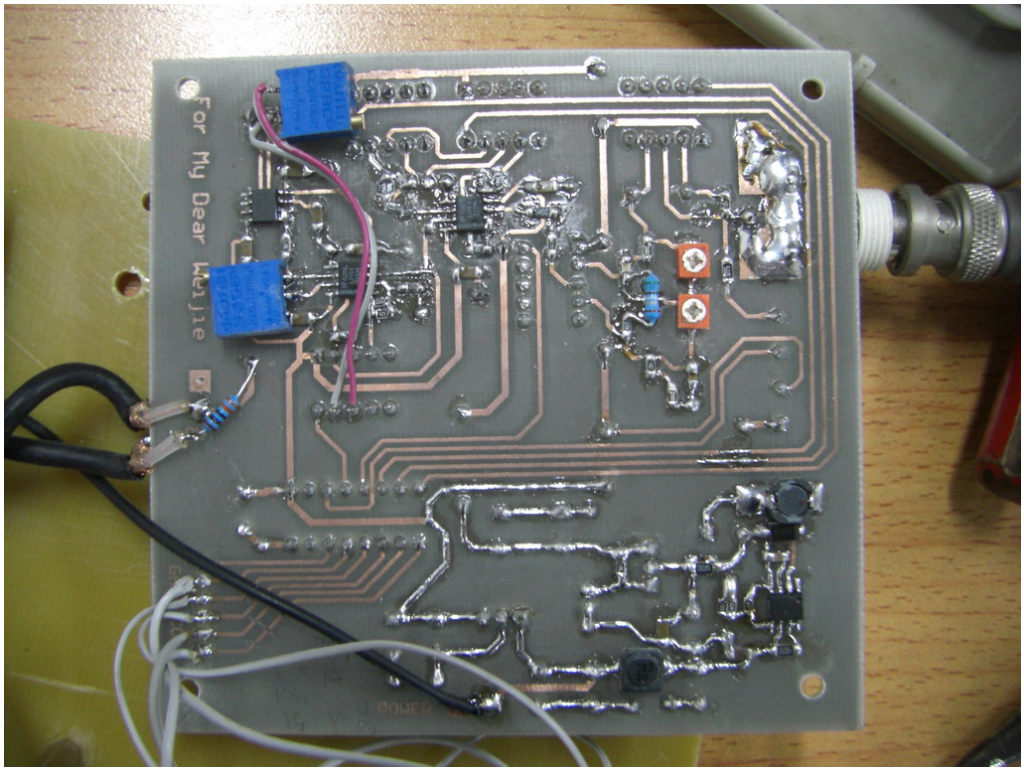


图 3：.程控放大器电路照片背面

2．高速 AD 转换与 FIFO 存储器电路

高速 AD 转换与 FIFO 存储电路 PCB 板图见图 4 所示，图 5 和图 6 为该板的实物图。从图 4 中可以看到其中体积最小管脚最密的为 AD 转换器 ADS830E 的焊盘，往左依次是 74LVC574 和 IDT7204 的焊盘，上面是 74F08 的焊盘。这个电路是示波器的咽喉要道，但不用调试！只是在焊接 ADS830E 时应注意一下焊接技巧，因为 ADS830E 的引脚比较多而且很密（引脚间距 0.65mm）所以焊接时容易因焊锡较多而造成短路，即然这样焊接时还不如给每个引脚上都镀上焊锡，先不要管他是否短路，焊在电路板上再说，焊好后用浸上松香的铜编织网将多余的焊锡吸掉就行，很简单的操作。后面的 mega32 的焊接方法相同，从图 6 照片中可以看出 ADS830E 旁边有很多松香，就是吸焊锡时残留的，虽然有些影响美观但对电路的性能没有影响，业余条件下还是可以接受的。

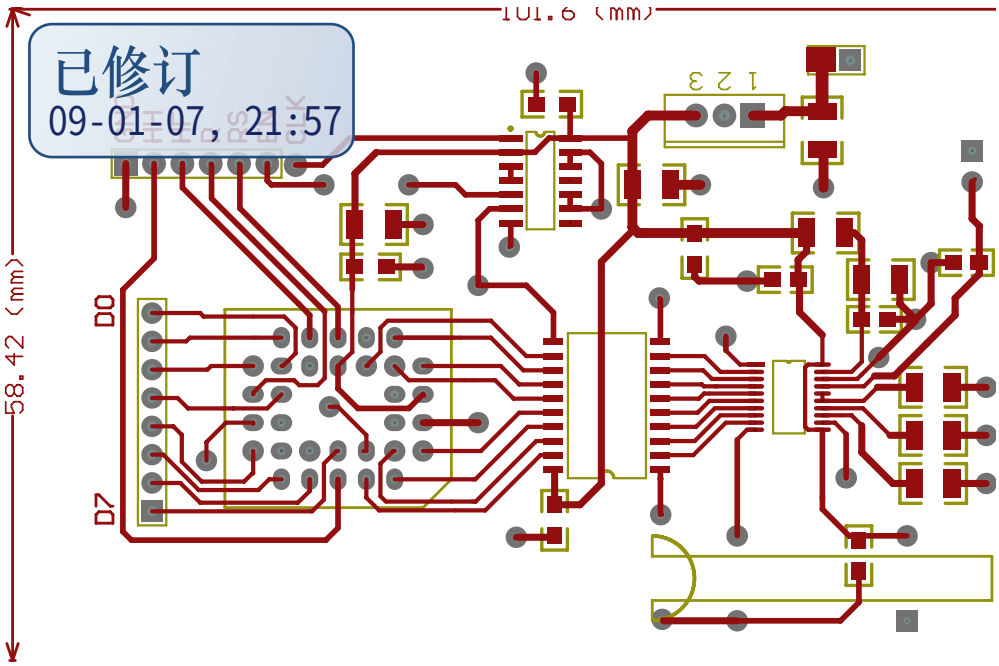


图 4：高速 AD 转换与 FIFO 存储电路 PCB 板图

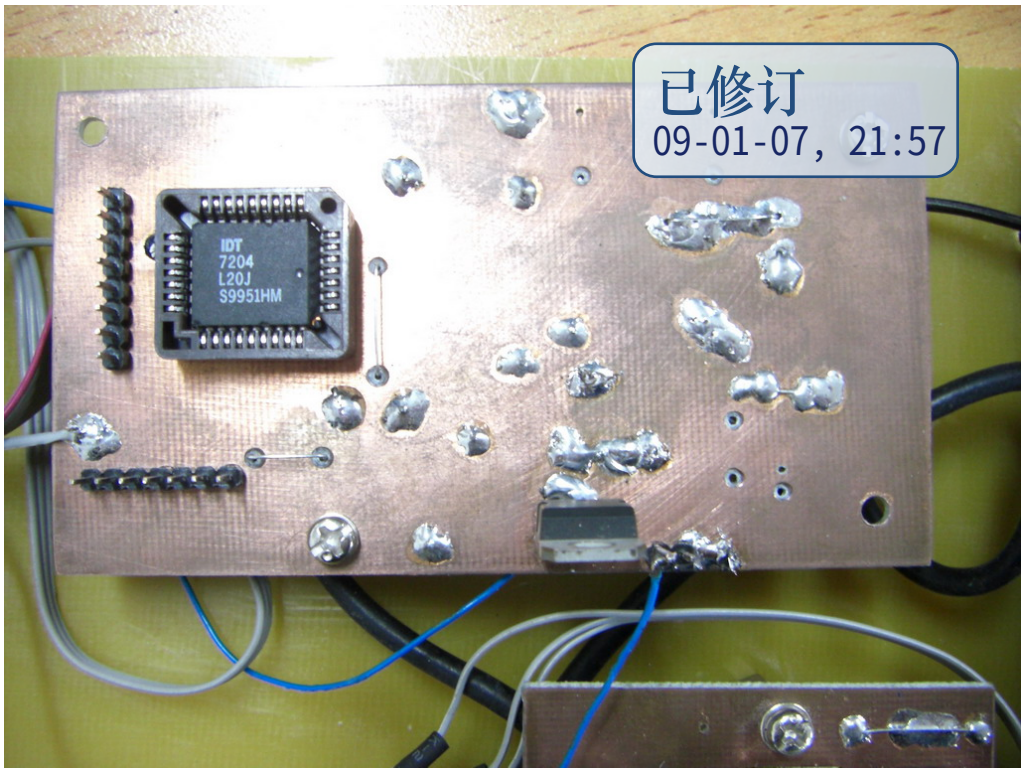


图 5：高速 AD 转换与 FIFO 存储电路照片正面

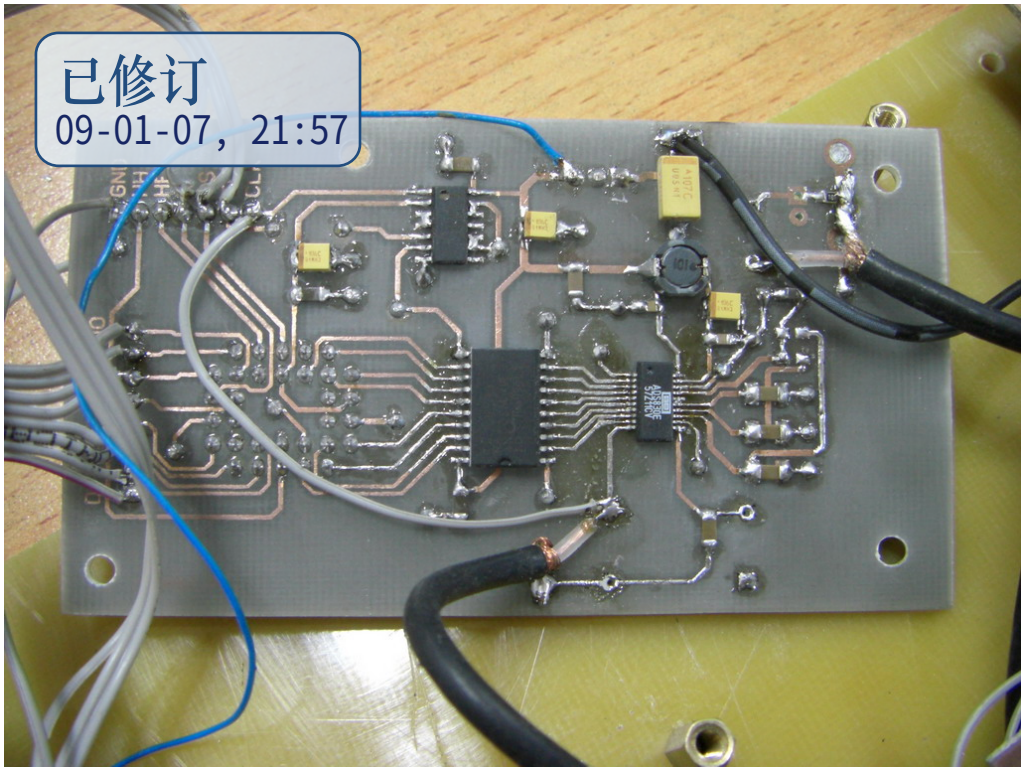


图 6：高速 AD 转换与 FIFO 存储电路照片背面

3．时钟产生电路

因为时钟产生电路的工作频率高达 60MHz，为使其工作稳定也使用双面板做，一面铜箔全部接地对高频信号进行屏蔽减少辐射干扰。印刷板见图 7 所示，图 8 为实物照片。因为该电路的最高工作频率为 60MHz，一般的 74LS 系列或 74HC 系列数字逻辑电路最高只能工作在 40MHz 左右，不能满足该电路的要求，所以只能选用速度等级更高的 74F 系列和 74AC 系列，相对来说 74F 系列更好买，所以本电路中的两个与 60MHz 信号相关的芯片 74F74 和 74F151 都选择 74F 系列，AD 转换电路中的与门因为也有可能工作在 60MHz（最快的两个扫速），所以也选用了 74F 系列中的 74F08。由于该电路板上全部为数字电路，所以只要元件选择正确且安装无误不用调试即可正常工作，所以就不多说了。

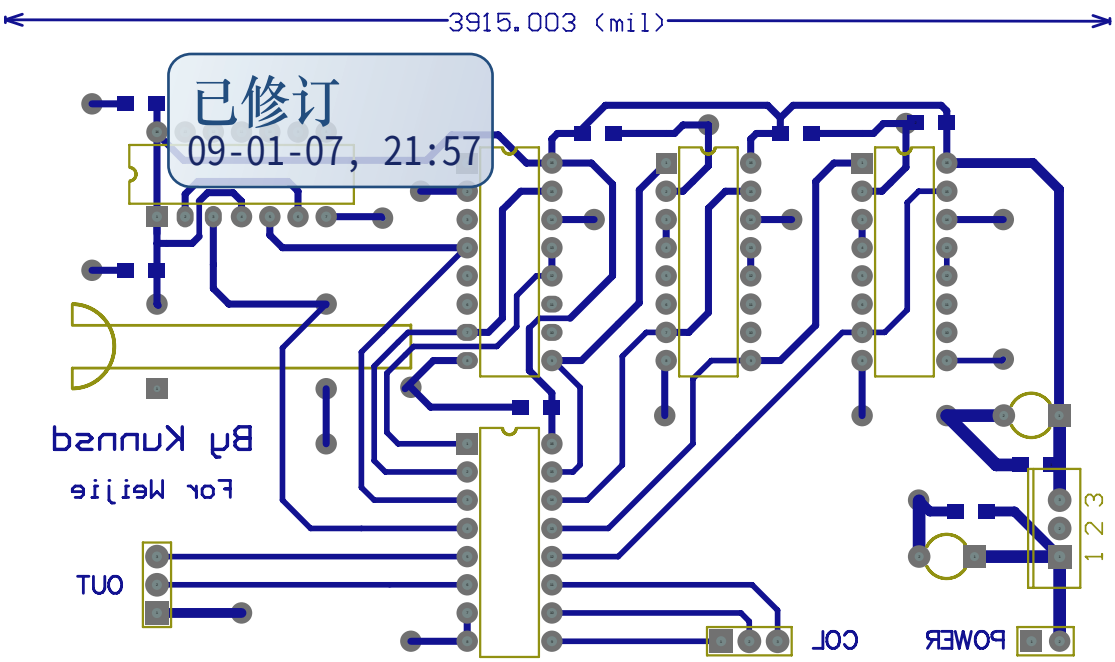


图 7：时钟产生电路 PCB 板图

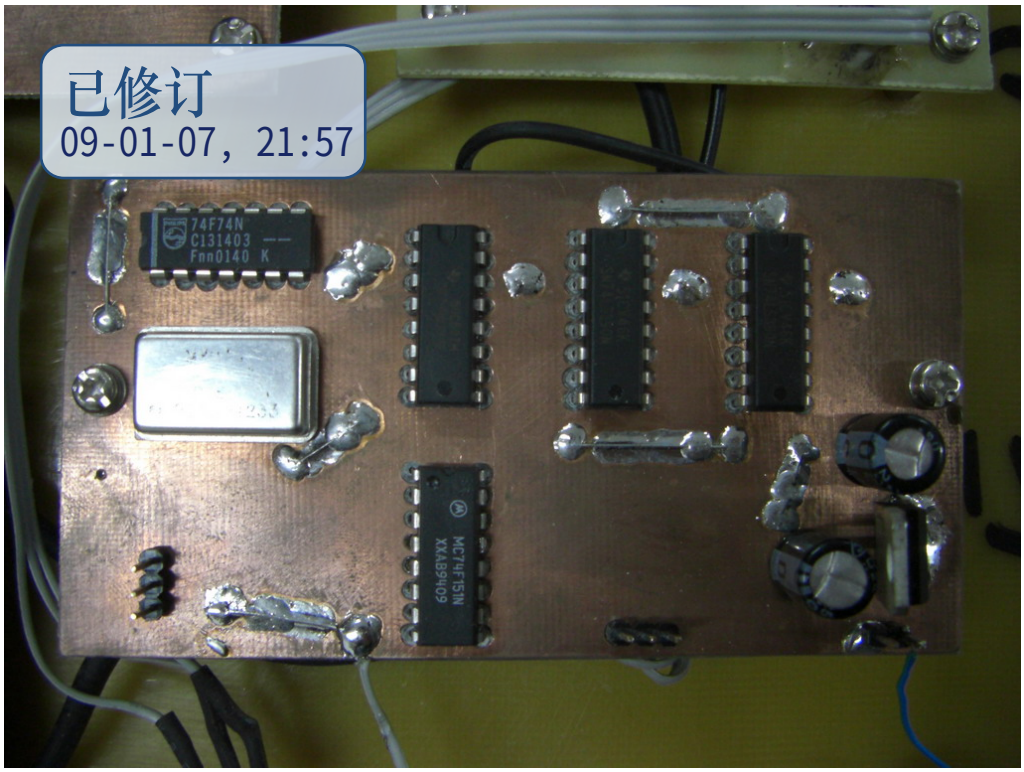


图 8：时钟产生电路照片

4．MCU2 控制电路

MCU2 控制电路板其实就是一个 mega32 的最小系统板，板子上有复位电路、振荡器等保证 mega32 正常工作所需的基本电路和 ISP 程序下载接口，32 个 IO 口全部引出，与各部分模块相连，见整体图。图 9 为 MCU2 控制电路的 PCB 板，图 10 和图 11 为做好后的照片，

从图 10 中可以看出 mega32 所使用的晶振为 18MHz，如硬件电路介绍中所述用 18MHz 晶振是为了提高屏幕的刷新率，增强示波器的实时性。这块板子的制作没有什么特殊要求，只是焊接 mega32 时用焊接 ADS830E 的方法即可，就不多说了。LCD 显示器引线采用扁平排线，所以要用接口连接。为了试验方便将 LCD 模块的接口电路单独做成一块 PCB 板，没有做在这块板上，见图 12 所示，图 13 为 LCD 适配器电路的照片。因为我做的这个示波器中的电路都是一步一步实验确定的，途中一个功能电路用好几种方案，最后确定最佳电路。所以电路都是做成模块的，这样的话便于组合实验，当确定一整套最佳方案后再将所有的模块做在一块电路板上。

图 14 为将 LCD 模块与 LCD 接口电路连接起来的照片，在电路都固定好之后，LCD 模块的接口电路是藏在 LCD 模块底下的，在整体的照片中是看不到的。

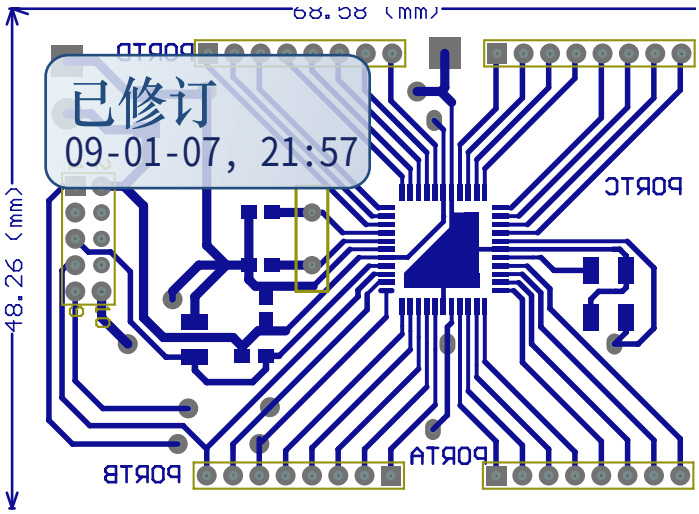


图 9：MCU2 控制电路 PCB 板图

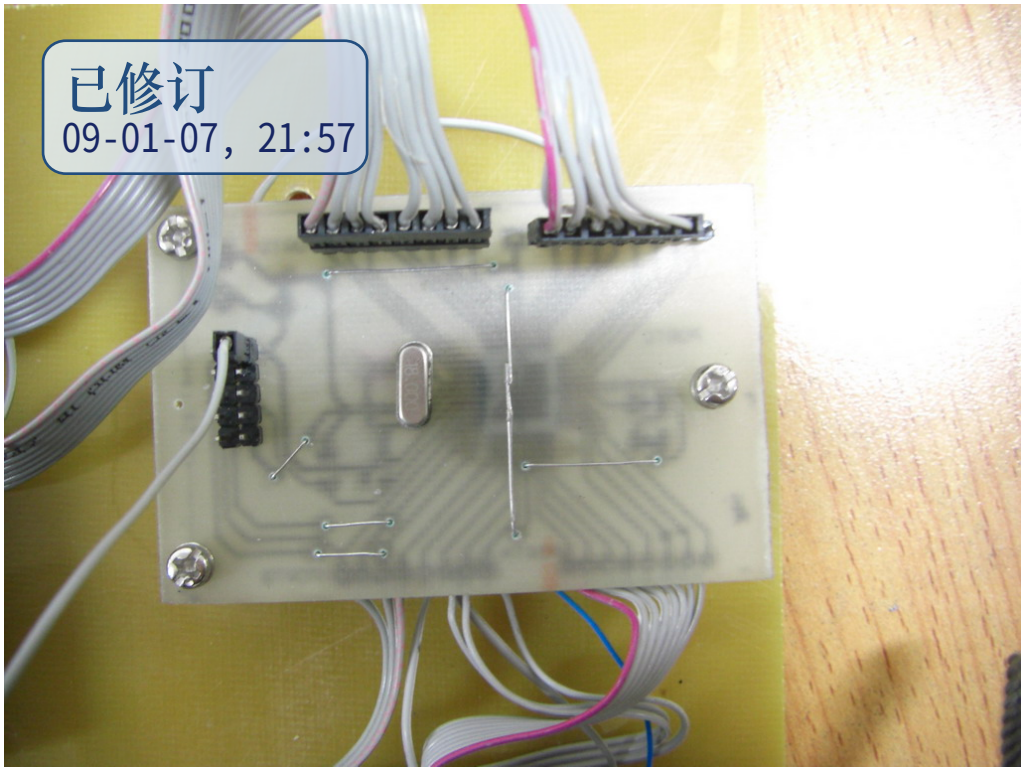


图 10：MCU2 控制电路照片正面

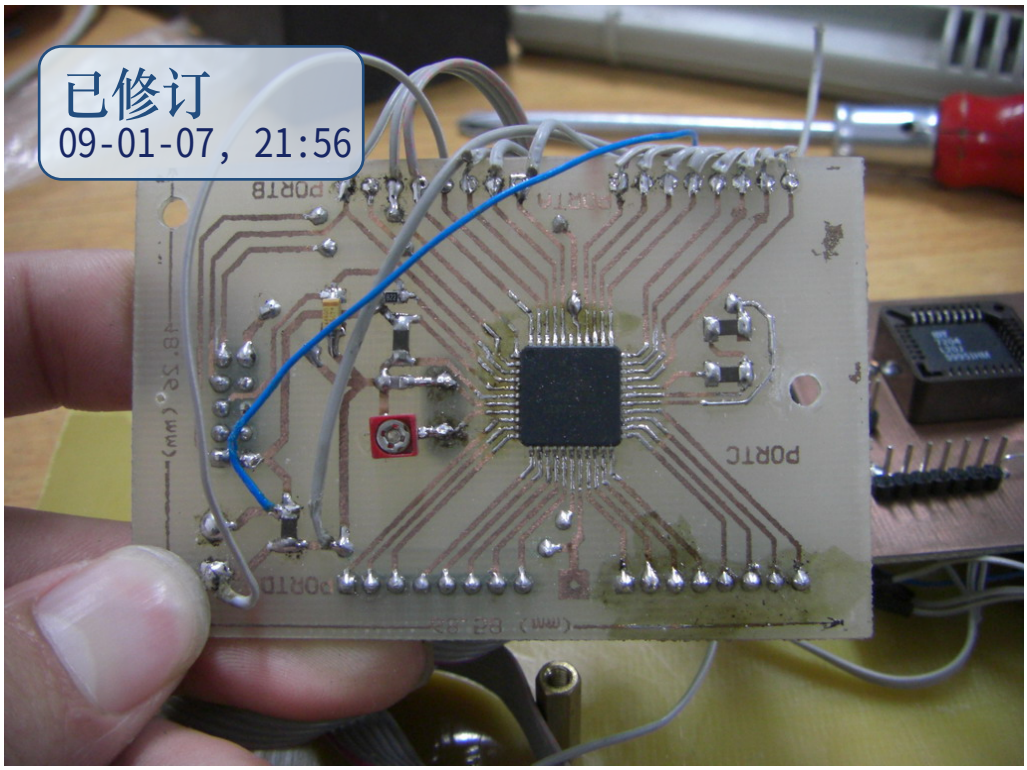


图 11：MCU2 控制电路照片背面

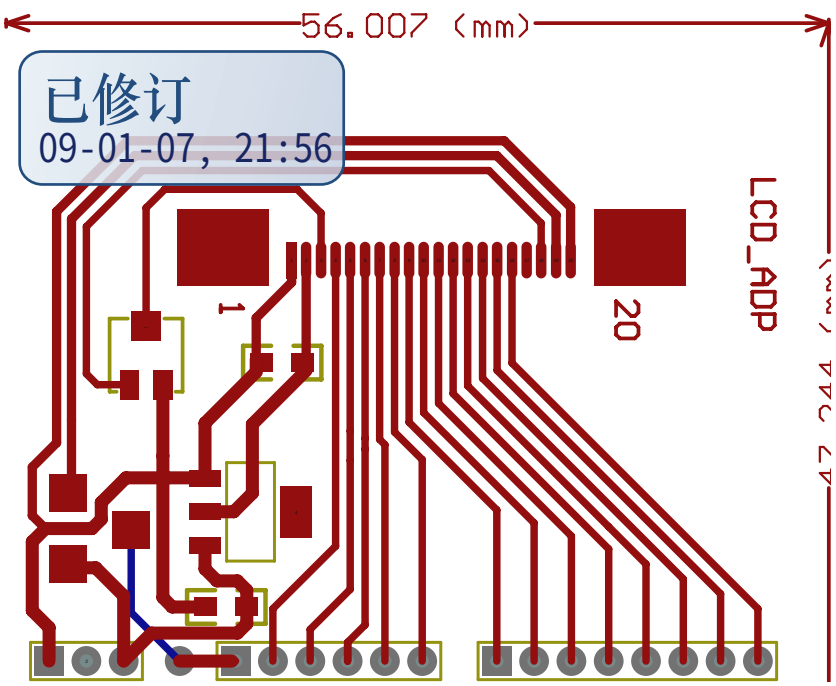


图 12：LCD 适配器电路 PCB 板图

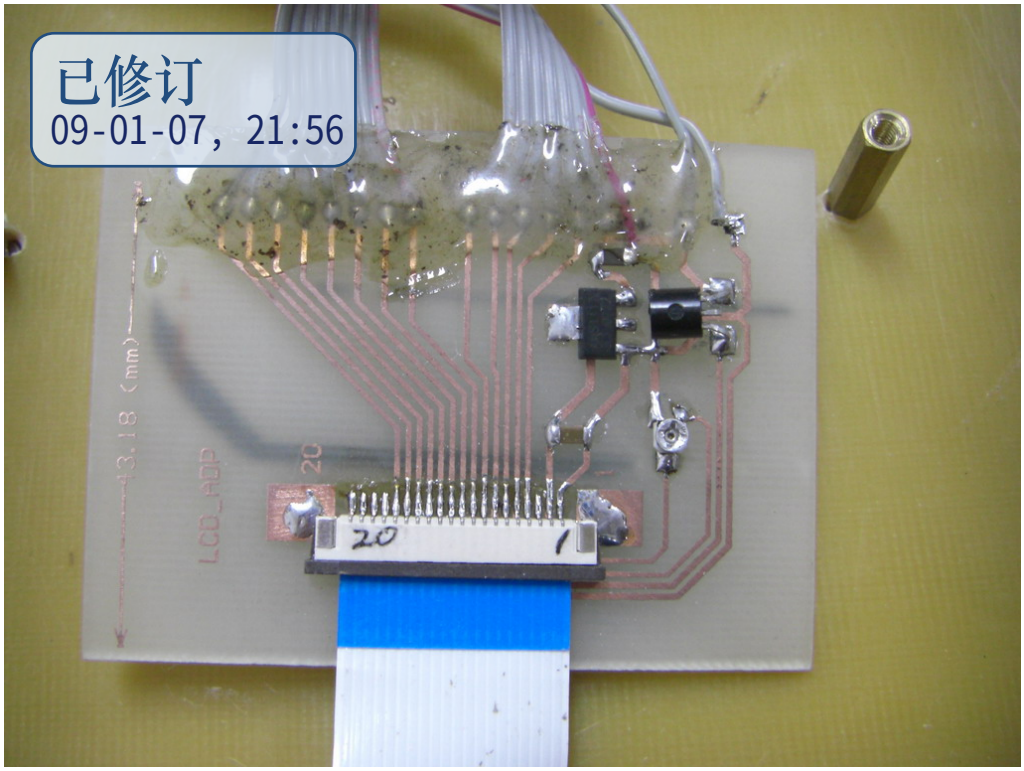


图 13：LCD 适配器电路照片

图 14：LCD 适配器与 LCD 连接照片

5．MCU1 控制电路与整形电路

MCU1 控制电路的 PCB 板图见图 15，mega8 单片机使用双列插 DIP 封装，这种封装的好处容易焊接，但占用板子的面积大，如果想要做成示波表，最好将 mega8 换成贴片封装的，以节省印刷板的面积，如果要扩展更多的按键，那么最好使用一片 mega16-16AI，因为 mega8 一共有 23 个 IO 口，在本电路中基本将 IO 口用完了，所以还是换一个 32 个 IO 口的 mega16 比较方便，成本增加不了几块钱，当然印刷板就得重新设计了。在安装这块电路板时有一个细节要注意，就是将所有元件焊完后在焊接场效应管 2sk192，而且焊接时要将烙铁从插座上拔下，以免烙铁上有感应静电，因为 2sk192 的栅极输入电阻极高，而且结电容很小，所以一点点的静电感应就可以让结电容上产生足以击穿栅极的电压，所以在储存和焊接时要特别注意。平时存放内部没有保护二极管的场效应管时应存放在金属盒中或用金属箔包裹后存放，尤其是在空气干燥的北方。

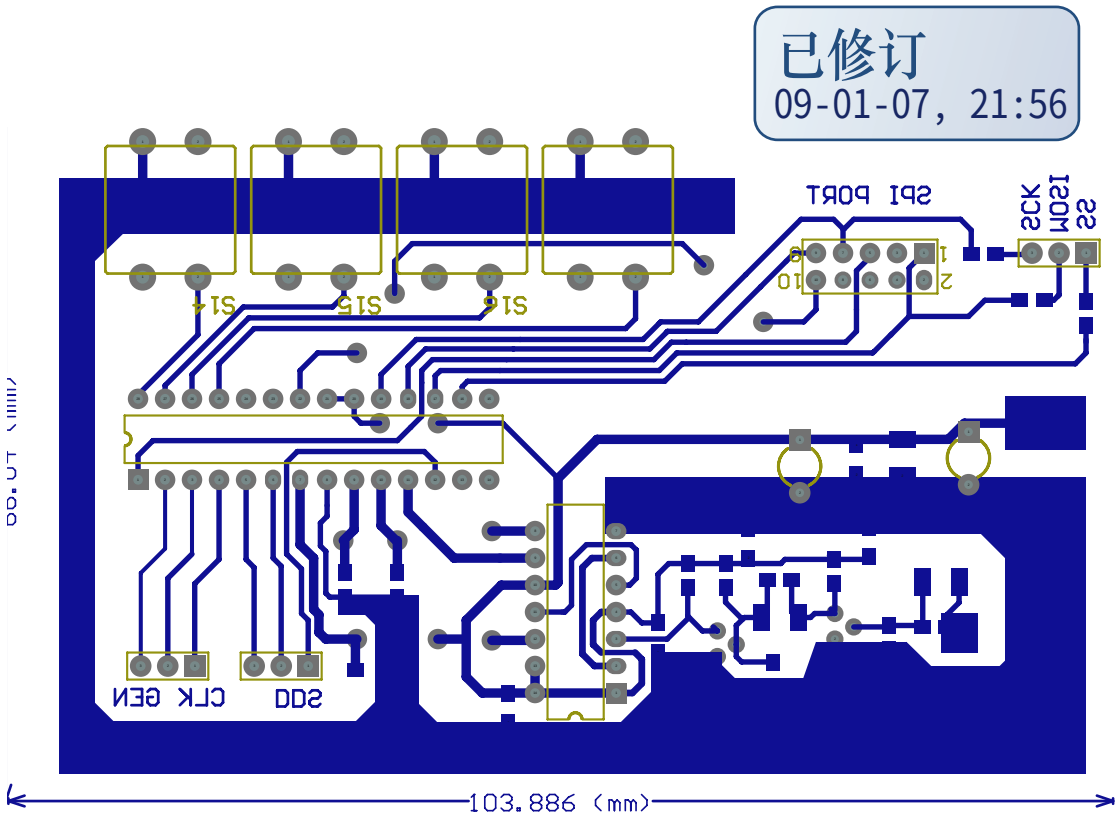


图 15：MCU1 及整形电路 PCB 板图

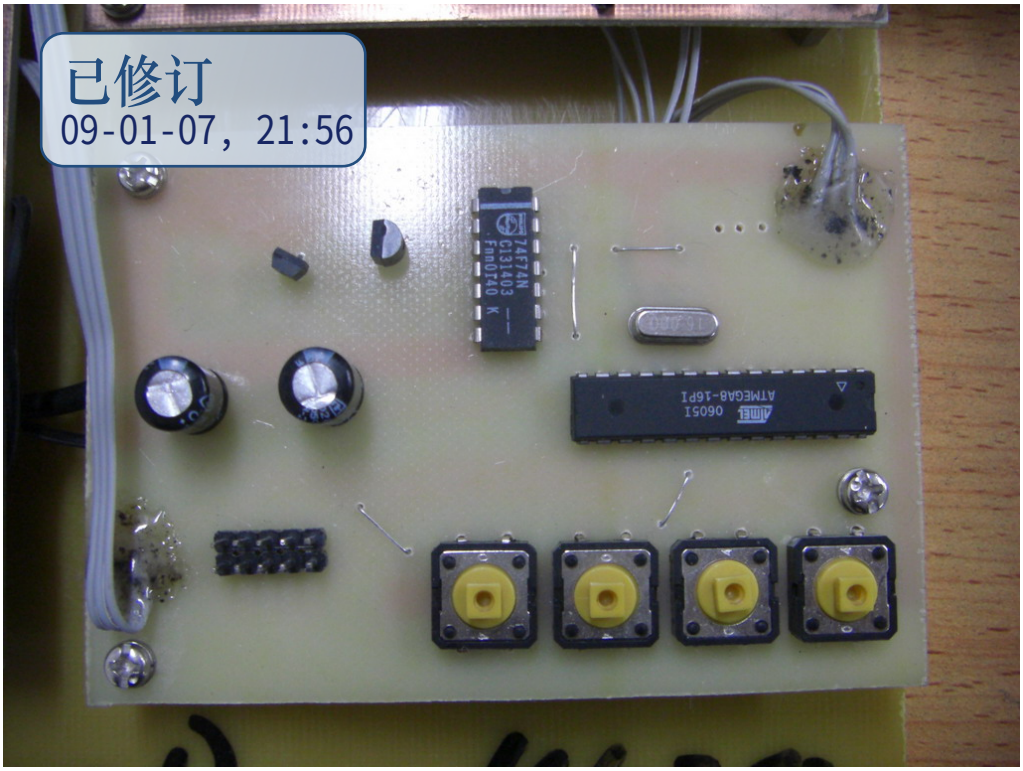


图 16：MCU1 电路照片

三．电路调试

1. 在焊接程控放大电路时先焊板上的电源部分，首先是MC34063电路，该电路只要元件选择正确一般不会出现问题，焊完MC34063部分后给电源输入端接上8V的电源，

测量C51两端应改为-8.3V左右，输入空载电流不超过10mA，否则应检查电路连接。测量正常后断开电源，安装LM337、LM317、LM7805以及外围元件，装好后通电，先调整负电源。调节Rw3，使LM337输出端C55两端的电压为-5V，调整Rw2使LM317输出端C46两端的电压为+5V即可，断开电源继续安装其他元件。

2. 在电路焊接完毕后，将程序分别写入两片单片机，接上显示器通电，一般显示器只会背光亮而无显示，或显示全黑点（图17和图18所示），这是因为显示器的显示偏置电压不正常，要通过调节LCD转接板上的电位器Rw_lcd调整，慢慢旋转Rw_lcd的螺口直到LCD显示正常，正常的显示应该为实测波形图那样。

图17：LCD亮度偏置过低导致“无显示”

图18：LCD亮度偏置过低导致“黑屏”

3. 将以上调整好后，本示波器就能正常显示了，这一步要做的就是让它正确显示，进入示波器界面后屏幕的右边应该能看见水平扫速、电压灵敏度等各种参数，显示可能不正常但应该有显示，否则说明两片单片机之间的通信不正常，要检查三根SPI通信线是否连接正确，如果有参数显示继续往下走。下来是调整示波器的基线，未调整的基线一般不会在水平中线位置，将程控放大电路的输入端对地短路，这时会在LCD屏上看到一条横线（如图19所示），如果横线在最上边或最下边说明基线电压偏离太远，这时调节程控放大器电路板上的Rw1，使其回到水平中线位置（如图20所示），此时测量程控放大器的输出端对地的直流电压应改为2.5V，因为AD转换器的输入电压范围为1.5V至3.5V，输入幅度为2Vpp，所以基线电压应该为AD转换器的输入中点电压。在调节好基线电压后，按4个功能键看各参数的变化是否正常，否则检查各控制线是否连接正常，我曾将控制时钟的4根线高低位接反导致屏幕上显示的最高水平扫速而实际是最低水平扫速，程控放大器也一样，接错会导致垂直电压灵敏度混乱，由于各电路板之间的连线比较多，所以在连接各块电路板时一定要细心谨防接错。

图19：基线偏离

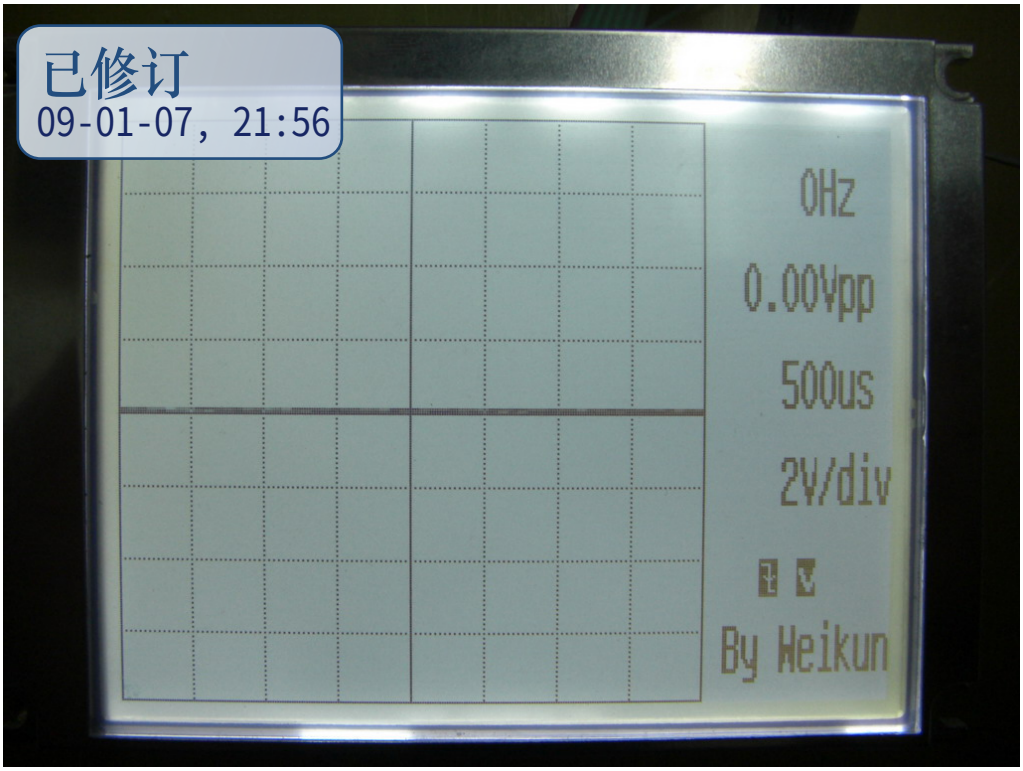


图20：基线正常

4. 在进行完上述的调整之后这个示波器就基本能够使用了，但在进行较高频率测量时（高于100kHz）还需要调整输入衰减电路中的边沿补偿电容，方法是从示波器的探

头输入300kHz有效值为1V的方波信号，在未进行补偿调整之前一般会出现两种情况，分别如图21和图22所示，图21所示的波形显示的情况是补偿过量，图22所示的波形显示的情况是补偿不足，在这种情况下仔细调整程控放大器电路中的两个可变电容C2和C3，直到波形正常，如图23所示。总装起来见图24。

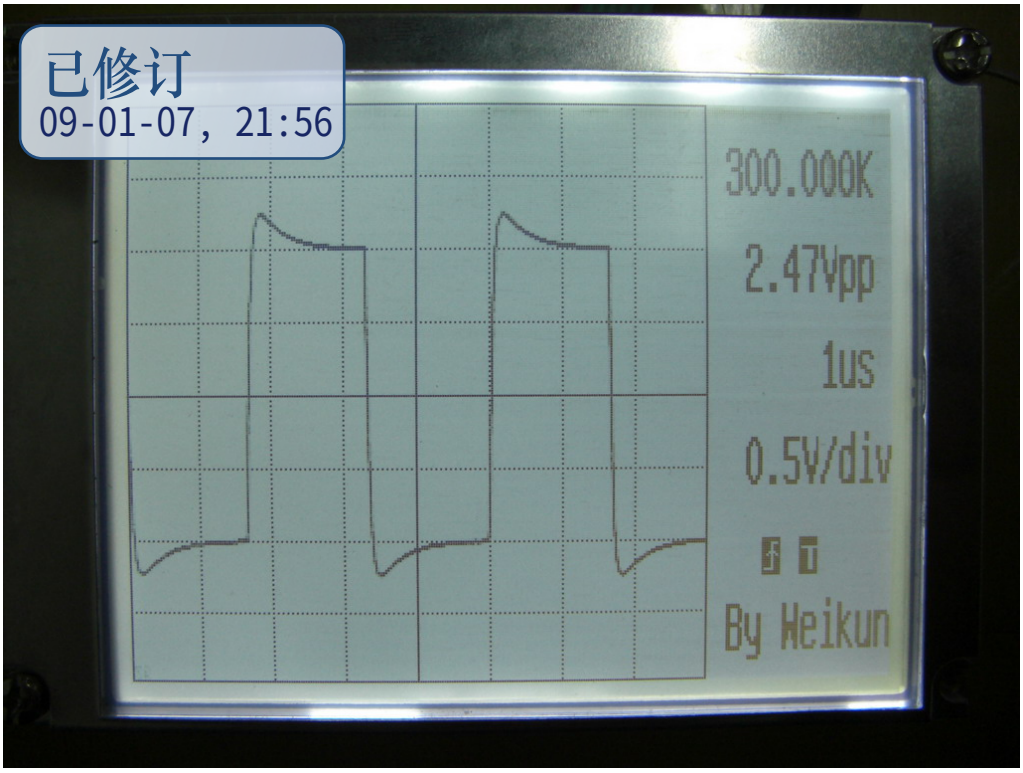


图21：补偿过量

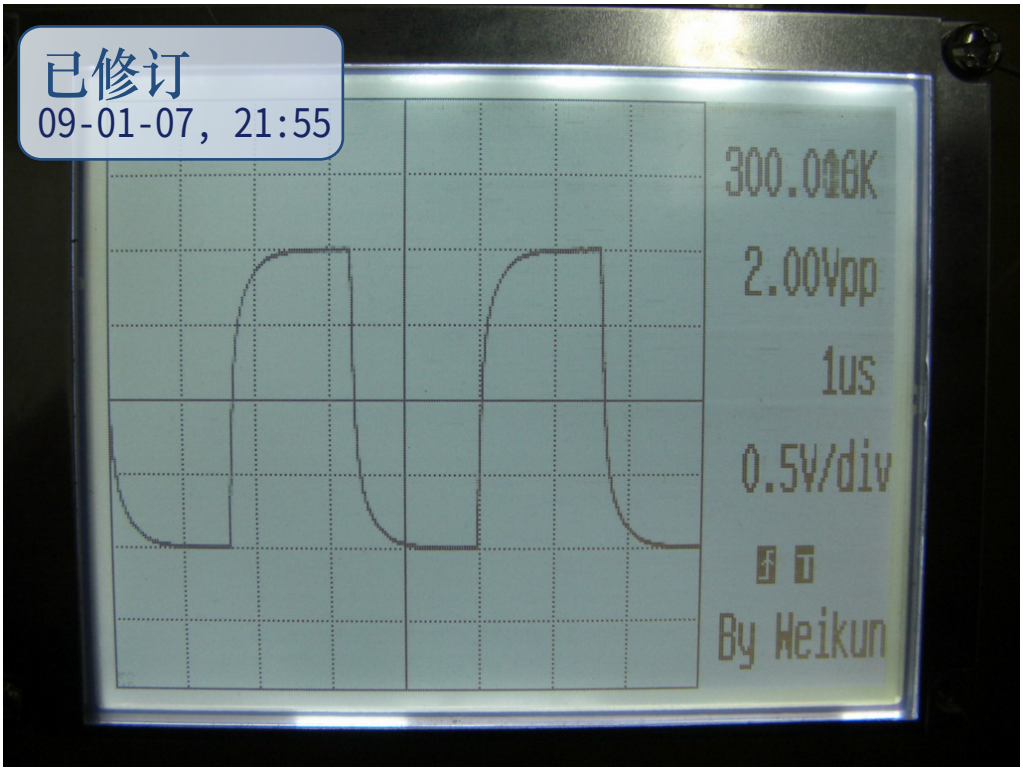


图22：补偿不足

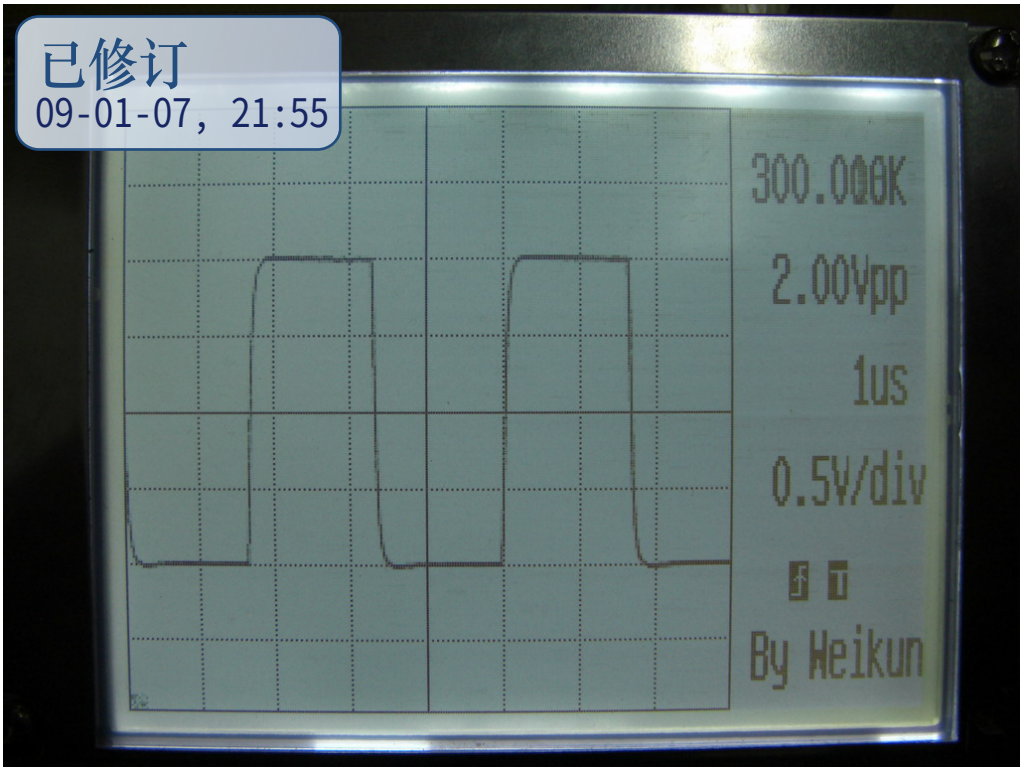
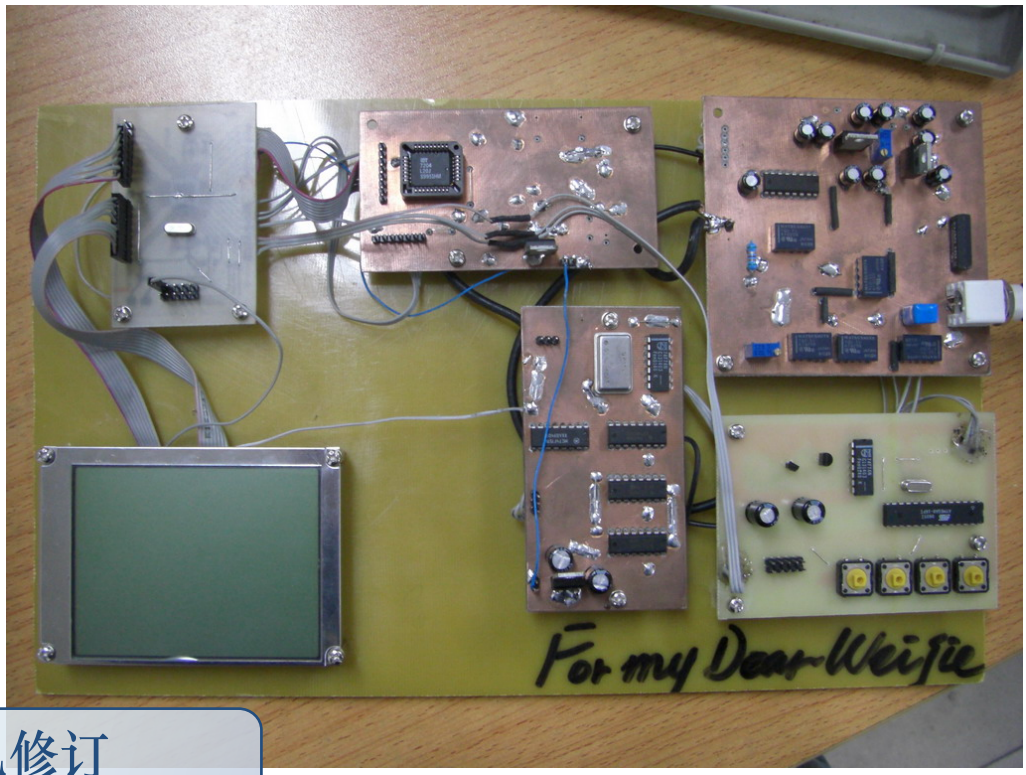


图23：补偿正常

5. 对MCU1的振荡频率微调

这是这台示波器最后需要调整的电路了。MCU1承担着测量频率的任务，测频的原理

是数规定时间内脉冲的个数，所以这个“规定时间”就是测量频率的基准时间，它的准确性将直接影响频率测量的精度。测量频率的基准时钟由MCU1的晶体振荡器提供，所以在购买晶振时一定要选择质量好精度高的优等品，在装调好其他电路后还要对MCU1的晶体振荡器进行微调，具体办法是用一个频率精度比较高信号源（我使用的是一台DDS信号源）产生一个待测信号给示波器的输入端，调整合适的扫速和垂直灵敏度，在显示器上会显示出波形和频率值，频率值的最低位一般都是会有一点偏差的，这时微调MCU1电路中的C24，使示波器的显示频率值与信号发生器的输出频率值相等即可，为了保证调整的可靠性最好多用几个频率跨度比较大的值进行调整。



已修订
09-01-07, 21:55

图24：装好后的效果

写到这里,关于这款示波器的制作已经叙述完毕了，对于这个示波器的性能我并不满足，我还会做它的第二版、第三版，有什么进展我还是会毫不保留的与大家分享，我提一些简易，也许对大家自制示波器有用：如果本电路中所有集成电路全部用小型的贴片封装，时钟产生电路的那些通用数字芯片用一片CPLD代替，继电器换用宽带的模拟开关那么这个示波器所有的电路都能做到像LCD显示器那么大的一块双面板上，因为不用继电器所以更省电,整个电路装进一个合适的壳子中用两节锂电池供电不就成了示波表吗？通过修改程序还可以加入更多实用功能,比如用一个定时器做PWM输出控制LCD显示

器的背光亮度，通过扩展外部存储空间或换用具有更大容量RAM的单片机如Mega128等实现波形存储回放，或在对实时性要求不太高的情况下对采样的数据进行FFT运算近似分析频谱等。更多功能的实现有待与广大电子爱好者的共同努力，我的这个不太完善的作品只想给广大的电子爱好者们一个信心——示波器也是可以自己做的，希望不久的将来会有更加完善与更加廉价的DIY示波器作品的出现，使每个电子爱好者都能有一台实用的DIY示波器！

魏坤

2008 11 15