目录

基于单片机的波形发生器的设计

学　　生:李利刚
指导老师:李　敏
（黄冈职业技术学院）

摘要：函数发生器是一种用于产生标准信号的电子仪器，它广泛用于工业生产、科研和国防等各个领域中，所以论文选题具有一定的实用意义。本文介在绍了函数发生器的基本概念及原理的基础上，采用 AT89C51 单片机为核心，完成了简易的 DDS 函数发生器的硬件设计和软件编程，并通过调试实现了其功能和主要技术指标。在系统的硬件部分，设计了由单片机最小系统为核心、通过接口设计，扩展了 DAC 转换模块、按键和 LED 显示模块。其中，采用两片 DAC0832 实现了全数字化的频率合成器(简称 DDS)。系统的软件设计是在 keil uVision4 的集成开发环境下，采用 C 语言完成了应用系统软件编程，包括主程序、产生四种常用信号的程序、按键功能和显示子程序等电脑模块；模块化的编程使得程序具有可读性和易于维护的特点。

关键词：信号发生器　单片机　keilc51

Based on SCM waveform generator design

Abstract：Function generator is used to produce a standard signal electronic instrument, it is widely used in industrial production, scientific research and national defense, and other fields, so the thesis has certain practical significance. This paper in shaoxing interface the function generator of basic concepts and principles, and on the basis of the AT89C51 microcontroller as the core, the completion of the simple DDS function generator hardware design and software programming, and through the debugging realized its function and the main technical indexes. In the system hardware part, designed by single chip minimize system as the core, through the interface design, expanded the DAC conversion module, keys and LED display module. Among them, the two pieces of DAC0832 realized the full digital frequency synthesizer (hereinafter referred to as DDS). The software design is in the system of the keil uVision4 integrated development environment, using C language completed application system software programming, including the main program, produce four common signal procedures, key functions and display subroutines computer module; Modular programming makes the program has a readable and easy maintenance characteristic.

Key words：Signal generator Single-chip microcomputer keilc51

1 绪论

1.1 选题背景及意义

函数发生器又称信号源或振荡器，在生产实践和科技领域中有着广泛的应用。各种波形曲线均可以用三角函数方程式来表示。能够产生多种波形,如三角波、锯齿波、矩形波（含方波）、正弦波的电路被称为函数信号发生器。在通信、广播、电视系统，在工业、农业、生物医学等领域内，函数信号发生器在实验和设备检测中具有十分广泛的用途。

1.1.1 本课题的研究现状

函数发生器既可以构成独立的信号源，也可以是高性能网络分析仪、频谱仪及其它自动测试设备的组成部分。函数发生器的关键技术是多种高性能仪器的支撑技术，因为它能够提供高质量的精密信号源及扫频源，可使相应系统的检测过程大大简化，降低检测费用并极大地提高检测精度。美国安捷伦生产的 33250A 型函数任意波形发生器可以产生稳定、精确和低失真的任意波形，其输出频率范围为 1μHz～80MHz，而输出幅度为 10mVpp～10Vpp；该公司生产的 8648D 射频信号发生器的频率覆盖范围更可高达 9kHz～4GHz。国产 SG1060 数字合成信号发生器能双通道同时输出高分辨率、高精度、高可靠性的各种波形，频率覆盖范围为 1μHz～60MHz；国产 S1000 型数字合成扫频信号发生器通过采用新技术、新器件实现高精度、宽频带的扫频源，同时应用 DDS 和锁相技术，使频率范围从 1MHz～1024MHz 能精确地分辨到 100Hz，它既是一台高精度的扫频源，同时也是一台高精度的标准信号发生器。还有很多其它类型的信号发生器，他们各有各的优点，但是函数发生器总的趋势将向着宽频率覆盖、高频率精度、多功能、多用途、自动化和智能化方向发展。

目前，市场上的信号发生器多种多样，一般按频带分为以下几种：

超高频：频率范围 1MHz 以上，可达几十兆赫兹。

高频：几百 KHZ 到几 MHZ。

低频：频率范围为几十 HZ 到几百 KHZ。

超低频：频率范围为零点几赫兹到几百赫兹。

超高频信号发生器，产生波形一般用 LC 振荡电路。

高频、低频和超低频信号发生器，大多使用文氏桥振荡电路，即 RC 振荡电路，通过改变电容和电阻值，改变频率。

用以上原理设计的信号发生器，其输出波形一般只有两种，即正弦波和脉冲波，其零点不可调，而且价格也比较贵，一般在几百元左右。在实际应用中，超低频波和高频波一般是不用的，一般用中高频，即几十 HZ 到几 MHZ。用单片机 AT89C51,加上一片 DAC0832,就可以做成一个简单的信号发生器，其频率受单片机运行的程序的控制。我们可以把产生各种波形的程序，写在 ROM 中，装入本机，按用户的选择，运行不同的程序，产生不同的波形。再在 DAC0808 输出端加上一些电压变换电路，就完成了一个频率、幅值、零点均可调的多功能信号发生器的设计。这样的机器体积小，价格便宜，耗电少，频率适中。

1.1.2 选题目的及意义

函数发生器是一种经常使用的设备，由纯粹物理器件构成的传统的设计方法存在许多弊端，如：体积较大、重量较沉、移动不够方便、信号失真较大、波形种类过于单一、波形形状调节过于死板，无法满足用户对精度、便携性、稳定性等的要求，研究设计出一种具有频率稳定、准确、波形质量好、输出频率范围宽、便携性好等特点的波形发生器具有较好的市场前景，以满足军事和民用领域对信号源的要求。

本次设计的主要目标是学习和运用单片机的 C 语言和汇编语言，利用单片机 AT89C51 和 8 位 D/A 转换芯片 DAC0832 共同实现正弦波，方波，三角波，锯齿波这四种常见波形的发生，并且可以接收外接键盘输入而在一定范围内改变频率。

在无标准函数发生仪器时，本设计可以作为简单的函数发生器使用。本次设计准备在成本较低廉的前提下完成，主要是用 AT89C51 单片机，DAC0832，性能指数都不是很高，所以对此信号源的基本要求是能发生几种常见的波形，正弦波，方波，三角波，锯齿波，并且能够在一定的范围内改变频率。通过该课题的设计掌握以 AT89C51 为核心的单片机系统的软硬件开发过程和基本信号的产生原理、测量及误差分析方法，同时掌握函数发生器系统的设计流程；培养我们综合运用所学的基本知识、基本理论和基本技能的能力，学习解决一般工程技术和有关专业问题的能力，学习工程设计和科学研究的基本方法，完成对所学知识的综合训练。

1.2 设计任务及要求

本设计采用 AT89C51 及其外围扩展系统，软件方面主要是应用 C 语言设计程序。系统以 AT89C51 单片机为核心，配置相应的外设及接口电路，用 C 语言开发，组成一个多功能信号发生系统。该系统的软件可运行于 Windows XP 环境下，硬件电路设计具有典型性。同时，本系统中任何一部分电路模块均可移植于实用开发系统的设计中，电路设计具有实用性。

1.2.1 设计的基本要求

（1）功能要求

1. 能产生正弦波、方波、三角波、锯齿波等 4 种周期性波形，并且可通调节变形成其它相关波形。
2. 用键盘输入可生成正弦波的基波及各次谐波单独的波形，也可生成基波和各次谐波线形组合的波形。
3. 输出波形的频率范围为 1MHz～1Hz；可以通过键盘输入粗调频率。
4. 输出波形幅度范围为 0～5V（峰-峰值），可调整。
5. 具有显示输出波形类型、及其粗调频率和幅度的功能。

1.2.2 本文结构安排

全文共分为 6 章，第 1 章绪论（介绍设计的研究现状、选题意义及设计的任务与要求）；第 2 章系统总体的设计原理；第 3 章系统的硬件设计；第 4 章系统的软件设计；第 5 章系统的调试与测试；第 6 章总结与展望

2 函数发生器系统设计

2.1 设计方案的比较

函数发生器的设计方案可用多种方案来完成。在设计前对各种方案进行了比较：

方案一：用差分放大电路实现三角波到正弦波以及集成运放组成的电路实现函数发生器。波形变换的原理是利用差分放大器的传输特性曲线的非线性，传输特性曲线越对称，线性区域越窄越好;三角波的幅度应正好使晶体接近饱和区域或者截至区域。

方案二：用二极管折线近似电路以及集成运放组成的电路实现函数发生器。根据二极管折线近似电路实现三角波——正弦波的变换频率调节部分设计时，可先按三个频率段给定三个电容值：1000pF、0.01mf、0.1μF 然后再计算 R 的大小。手控与压控部分线路要求更换方便。为满足对方波前后沿时间的要求，以及正弦波最高工作频率（1MHz）的要求，在积分器、比较器、正弦波转换器和输出级中应选用 Sr 值较大的运放（如 LF353）。为保证正弦波有较小的失真度，应正确计算二极管网络的电阻参数，并注意调节输出三角波的幅度和对称度。输入波形中不能含有直流成分。

方案三 ：利用单片机 AT89C51 和 8 位 D/A 转换芯片 DAC0832 共同实现正弦波，方波，三角波，锯齿波这四种常见波形的发生，并且可以接收外接键盘输入而在一定范围内改变频率。

可行性分析：

上面三种方案中，方案一与方案二中三角波——正弦波部分原理虽然不一样，但是他们有共通的地方就是都要认为地搭建波形变换的电路图。而方案三利用单片机构成的应用系统有较大的可靠性。系统扩展、系统配置灵活。容易构成各种规模的应用系统，且应用系统有较高的软、硬件利用系数。单片机具有可编程性，硬件的功能描述可完全在软件上实现，而且设计时间短，成本低，可靠性高。

综上所述我们选择了第三种设计方案

2.2 系统模块设计

该函数发生器有以下几部分组成：（1）控制模块（2）按键及其显示模块（3）D/A 转换模块。

2.2.1 控制模块:

方案一：用单片 AT89C51 作为系统的主控核心。具有体积小，使用灵活的，易于人机对话和良好的数据处理，单指令周期和 35M 高速运算功能等优点。且单片机功耗低，价格低廉的优点。

方案二：用单片 AT89C51 作为系统的主控核心。具有价格低廉的优点，但处理速度较慢（1/12M）,AT89C51 是它的 35 倍。

方案三：用 FPGA 等可编程器件作为控制模块。FPGA 可以实现各种复杂的逻辑功能，密度高，速度快，稳定性好等许多有点。FPGA 在掉电后会丢失数据上电后须进行一次配置,因此FPGA在应用中需要配置电路和一定的程序。并且FPGA作为数字逻辑器件，竞争、冒险是数字逻辑器件较为突出的问题，因此在使用时必须注意毛刺的产生、消除及抗干扰性。

在次系统中，采用单片机作为控制比采用 FPGA 实现更简便。基于综合性价比，确定选择方案一.

2.2.2 按键及其显示模块:

方案一：采用传统的独立式按键；用传统的 LED 段选位选的方式进行波形的切换及显示。这种方式占用系统资源较多，并且效率低，程系编写大量而复杂。

方案二：为了提高单片机的资源利用率和运行的效率,按键显示部分我们直接使用 zlg7289 扩展键盘，键盘与单片机连接。zlg7289 芯片与单片机之间通信方便，而且由 zlg7289 对键盘进行自动扫描，可以去抖动，充分的提高了单片机的工作效率。

在次系统中,我们直接采用 zlg7289 扩展键盘实现更简便,确定选择方案二。

2.2.3 波形产生模块

方案一：使用锁相环通过分频运算实现频率的步进，这种方案频率稳定度高，但程控比较困难，而且步进范围过大，鉴于锁相环技术比较复杂，没有采用这种方案。

方案二：使用专用函数发生电路，如ICL8038 或MAX038，通过D/A转换调整函数发生器控制电压实现频率的控制，这种方案可以使频率连续可调，省却了波形转换电路，但控制电压与频率的变化不是严格的线性关系，如果不加频率负反馈则频率无法稳定准确，加上频率负反馈将使电路大大复杂，稳定度也会下降，而且如果要实现比较大的带宽,就需要不断更换振荡电容,电路复杂程度进一步增加。为避免调试困难，没有采用这种方案。

方案三：使用单片机的定时器设置定时时间，每半个周期对 I/O 口取反一次，从而实现频率输出。这种方案虽然在高频频段误差比较大，但是编程简单控制容易，权衡以上利弊，我们选择了方案三。

2.2.4 D/A 转换

单片输出的是数字信号，必须通过 D/A 转换后才能模拟信号。

方案一：采用 D/A 转换器 AD7543。AD7543 是一种串行的 D/A 转换器，与单片机之间的连线少，布线方便，而且又是 12 位的 D/A 转换器，精度高。但串行数据传输速度慢,当频率较高时,必须减少每周期输出的点数,这将会导致阶梯现象更加明显,因此,此方案不宜使用。

方案二：采用 DAC0832。这是 8 位的并行 D/A 转换器，转换速度快。

方案三：采用 2 片 DAC0832。由其中一芯片的输出电压作为另一芯片的参考电压，这样就可以方便的控制最大输出电压。

若采用方案二，在输出电压较低的情况下，比如为 1V 时，输出的最大电压只有参考电压的 1/5，这将会使精度降低，而方案三刚好可以解决这个问题，因此，本系统选择了方案三。

DAC0832 芯片介绍

DAC0832 是美国国家半导体公司生产的一种 8 位分辨率、双通道 A/D 转换芯片。由于它体积小，兼容性，性价比高而深受单片机爱好者及企业欢迎，其目前已经有很高的普及率。学习并使用 DAC0832 可是使我们了解 A/D 转换器的原理，有助于我们单片机技术水平的提高。

DAC0832 具有以下特点：

- 8 位分辨率；
- 双通道 A/D 转换；
- 输入输出电平与 TTL/CMOS 相兼容；
- 5V 电源供电时输入电压在 0 到 5V 之间；
- 电流建立时间 1μS；
- 一般功耗仅为 15mW；
- 8P、14P—DIP（双列直插）、PICC 多种封装；
- 商用级芯片温宽为 0° C 到 +70° C，工业级芯片温宽为-40° C 到 +85° C；

芯片接口说明：

- CS 片选使能，低电平芯片使能。
- CH0 模拟输入通道 0，或作为 IN+/-使用。
- CH1 模拟输入通道 1，或作为 IN+/-使用。
- GND 芯片参考 0 电位（地）。

- DI 数据信号输入，选择通道控制。
- DO 数据信号输出，转换数据输出。
- CLK 芯片时钟输入。
- Vcc/REF 电源输入及参考电压输入（复用）。

DAC0832 为 8 位分辨率 A/D 转换芯片，其最高分辨可达 256 级，可以适应一般的模拟量转换要求。其内部电源输入与参考电压的复用，使得芯片的模拟电压输入在 0 到 5V 之间。据有双数据输出可作为数据校验，以减少数据误差，转换速度快且稳定性能强。独立的芯片使能输入，使多器件挂接和处理器控制变的更加方便。通过 DI 数据输入端，可以轻易的实现通道功能的选择。

单片机对 DAC0832 的控制原理：

正常情况下 DAC0832 与单片机的接口应为 4 条数据线,分别是 CS、CLK、DO、DI。但由于 DO 端与 DI 端在通信时并未同时有效并与单片机的接口是双向的，所以电路设计时可以将 DO 和 DI 并联在一根数据线上使用。当 DAC0832 未工作时其 CS 输入端应为高电平,此时芯片禁用,CLK 和 DO/DI 的电平可任意。当要进行 A/D 转换时，须先将 CS 使能端置于低电平并且保持低电平直到转换完全结束。此时芯片开始转换工作，同时由处理器向芯片时钟输入端 CLK 输入时钟脉冲，DO/DI 端则使用 DI 端输入通道功能选择的数据信号。在第 1 个时钟脉冲的下沉之前 DI 端必须是高电平，表示启始信号。在第 2、3 个脉冲下沉之前 DI 端应输入 2 位数据用于选择通道功能 当此2 位数据为“1”、“0”时,只对CH0 进行单通道转换。当2位数据为“1”、“1”时，只对 CH1 进行单通道转换。当 2 位数据为“0”、“0”时，将 CH0 作为正输入端 IN+，CH1 作为负输入端 IN-进行输入。当 2 位数据为“0”、“1”时，将 CH0 作为负输入端 IN-，CH1 作为正输入端 IN+进行输入。到第 3 个脉冲的下沉之后 DI 端的输入电平就失去输入作用，此后 DO/DI 端则开始利用数据输出 DO 进行转换数据的读取。从第 4 个脉冲下沉开始由 DO 端输出转换数据最高位 DATA7，随后每一个脉冲下沉 DO 端输出下一位数据。直到第 11 个脉冲时发出最低位数据 DATA0，一个字节的数据输出完成。也正是从此位开始输出下一个相反字节的数据,即从第 11 个字节的下沉输出 DATA0。随后输出 8 位数据,到第 19 个脉冲时数据输出完成，也标志着一次 A/D 转换的结束。最后将 CS 置高电平禁用芯片，直接将转换后的数据进行处理就可以了。作为单通道模拟信号输入时 DAC0832 的输入电压是 0 到 5V 且 8 位分辨率时的电压精度为 19.53mV。如果作为由 IN+与 IN-输入的输入时，可是将电压值设定在某一个较大范围之内，从而提高转换的宽度。但值得注意的是，在进行 IN+与 IN-的输入时，如果 IN-的电压大于 IN+的电压则转换后的数据结果始终为 00H。

2.3 系统总体框图

本系统是以单片机 AT89C51 和 8 位 D/A 转换芯片 DAC0832 以及 zlg7289 键盘及显示共同实现正弦波，方波，三角波，锯齿波这四种常见波形的产生及显示相互切换的功能。

系统原理框图如图 2-1

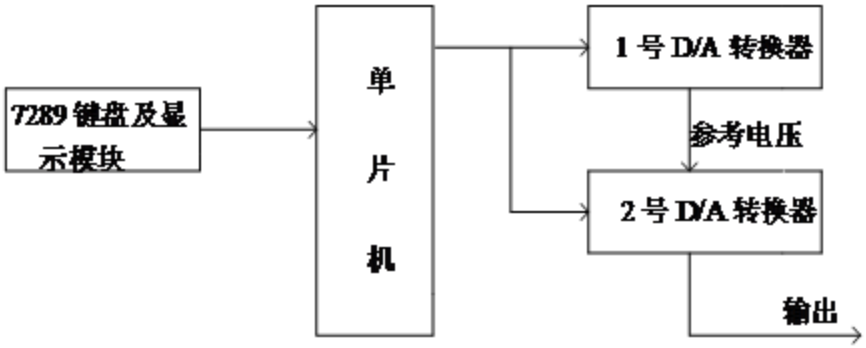


图 2-1　系统框图

2.4 理论分析

2.4.1 电路的理论计算

由图 3-6 可知到 U5 的输出将作为 DAC0832(2)的基准电压。

假设运放 U6 的输入为V_1；DAC0832(2)的内部的电阻为R_1。设 U5 的输入电压为V_2；DAC0832(2)的内部的电阻为R_2。下面进行讨论计算：

（1） U6 为一个反相比例器：

$$\frac{V_1}{R_1}=-\frac{U_{out1}}{R_{28}},U_{out1}=-\frac{R_{28}}{R_1}V_1$$

（2） U5 也为一个反相比例器：

$$\frac{V_2}{R_2}=-\frac{U_{out2}}{R_{26}},U_{out2}=-\frac{R_{26}}{R_2}V_2$$

这是 U4 的输入为$(U_{out1}+U_{out2})$，记为 V

由于$R_{25}=R_{14}$，这是 U4 实际上为为一个反响跟随器，即：

$$U_{out3} = -V = U_{out1} + U_{out2}$$

所以：

$$U_{out3} = -(-\frac{R_{28}}{R_1}V_1 - \frac{R_{26}}{R_2}V_2) = \frac{R_{28}}{R_1}V_1 + \frac{R_{26}}{R_2}V_2$$

2．本设计中的运放的连接的第二部分如图 3-6

如图 3-6 中的 U2 输入中的U_{out3}为图 3-7 中 U4 的输出。

对于本运放组成的电路分析采用叠加法：

设 U2 的输出为U_{out4}。

<1>当±12 电源全部接地时，有如下：

此时的运放实际为一个反相比例器：

$$R_1 = (R_{11} + R_{12右})//(R_{16} + R_{12左})$$

$$\frac{U - U_{out3}\dfrac{R_1 + R_{13}}{R_1 + R_{13} + R_{14}}}{R_{14}} = \frac{U_- - U_1}{R_9}$$

$$U_1 = -\frac{R_9(R_1 + R_{13})}{R_{14}(R_1 + R_{13} + R_{14})}U_{out3} \qquad (2\text{-}1)$$

<2>当-12V 和U_{out3}接地时：

$$R_2 = (R_{16} + R_{12右})//R_{13}$$

$$\frac{U_- - 12\dfrac{R_2}{R_2 + R_{11} + R_{12左}}}{R_{13}} = \frac{U_- - U_2}{R_9}$$

$$U_2 = -12\frac{R_9}{R_{13}}\frac{R_2}{R_2 + R_{11} + R_{12左}} \qquad (2\text{-}2)$$

<3>当 12V 和U_{out3}接地时：

$$R_3 = (R_{11} + R_{12左})//R_{13}$$

$$\frac{U_- + 12\dfrac{R_3}{R_3 + R_{16} + R_{12右}}}{R_{13}} = \frac{U_- - U_3}{R_9}$$

$$U_3 = 12\frac{R_9}{R_{13}}\frac{R_3}{R_3 + R_{11} + R_{12左}} \qquad (2\text{-}3)$$

所以综上所述：U_{out4}为U_1，U_2，U_3三者之和。

$U_{out4}=U_1+U_2+U_3$即：

$$U_{out4}=-\frac{R_9(R_1+R_{13})}{R_{14}(R_1+R_{13}+R_{14})}U_{out3}-12\frac{R_9}{R_{13}}\frac{R_2}{R_2+R_{11}+R_{12左}}+U_3+12\frac{R_9}{R_{13}}\frac{R_3}{R_3+R_{11}+R_{12左}}$$

$$=-\frac{R_9(R_1+R_{13})}{R_{14}(R_1+R_{13}+R_{14})}U_{out3}-12\frac{R_9}{R_{13}}(\frac{R_2}{R_2+R_{11}+R_{12左}}-\frac{R_3}{R_3+R_{11}+R_{12左}})$$

下面对U_{out4}的结果作一些辅助说明：

特例：当$R_2=R_3$时$U_{out4}=U_1=-\frac{R_9(R_1+R_{13})}{R_{14}(R_1+R_{13}+R_{14})}U_{out3}$　　（2-4）

以下进行代入数据的具体的分析：

于是对于将图 3-6 与图 3-7 及连在一起时，波形输出与调节部分的理论计算。

当单片机分别向 DAC0832（1）和 DAC0832（2）输入数据D_1和D_2时

$$U_{02}=-D_2\times VR=-12\times D_2/256 \qquad (2-5)$$

$$U_{01}=-D_1\times VR=-U_{02}\times D_1/256$$

$$U_0=-R_3/R_1\times U_{01}-U_{02}\times R_3/R_1 \qquad (2-6)$$

其中$R_1=R_3=10k\Omega$，$R_2=20k\Omega$，代入以上各式，得

$$U_0=U_{02}\times(D_1/128-1)/2 \qquad (2-7)$$

或者$D_1=128\times(U_0/U_{02}+1)$　　（2-8）

由（2-7）式可知，当D_1在 0～255 之间变化时，U_0可在$-\frac{U_{02}}{2}$～$+\frac{U_{02}}{2}$之间变化，即输出信号的峰峰值可由U_{02}控制。

该电路由 102 电位器串接 2 个 1kΩ电阻实现调节直流偏移，电位器触头在最右端和最左端时，电位器输出的电压分别为-5v 和+5v，电位器的电压与U_0通过一个加法器后，实现直流偏移的调节。

2.4.2 波形产生相关理论

DAC0832 是 8 位的 D/A 转换器件，其工作电压是 0—5V，当输入 00 数字量的时候，输出为 0V 电压；当输入 80 数字量的时候，输出为 2.5V 电压；当输入 FF

数字量的时候，输出为5V电压。单片机的I/O输出均为+5 V的TTL电平，因此产生的正弦波幅值为+2.5 V。将一个周期内的正弦波形等分为256份，那么第1点的角度为0°，对应的正弦值为2.5sin0°；第2点的角度为360°/256，对应的正弦值为2.5sin（360°/256）……，如此计算下去，将这些模拟量正弦值都转换为单极性方式下的数字量，得到一张按照点号顺序排列的正弦波波形数据表。而每次送到74LS373的八位数字量是根据查正弦波形数据表格而得到。

其实在计算正弦波形数据的时候，并不需要算出整个 0—2π区间的每一个值，而只需计算出0—$\frac{1}{2}$π中的值就行，其他区间的值都可以通过对0—$\frac{1}{2}$π中的值取不同的变换。比如$\frac{1}{2}$π—π的值可以和 0—$\frac{1}{2}$π值一一对应，而π—2π的值可以对0—π区间的值取反得到。计算值可以用C语言编程得到。

幅度公式为Y=2.5［1+sin($\frac{90}{64}N$)］（N=0,1,2……64）

相应的Y值数字化后的递增量δ=$\frac{5}{255}$≈0.0196

那么每一个点相对于起一个点的递增率为A= $\frac{Y2-Y1}{\delta}$（Y2当前的点，Y1为前一个点）

所以每一个点的数字量与递增率A成一一对应关系。

2.5 单片机软件开发系统

Keil C51是美国Keil Software公司出品的51系列兼容单片机C语言软件开发系统，与汇编相比，C语言在功能上、结构性、可读性、可维护性上有明显的优势，因而易学易用。用过汇编语言后再使用C来开发，体会更加深刻。Keil C51软件提供丰富的库函数和功能强大的集成开发调试工具，全Windows界面。另外重要的一点，只要看一下编译后生成的汇编代码，就能体会到Keil C51生成的目标代码效率非常之高，多数语句生成的汇编代码很紧凑，容易理解。在开发大型软件时更能体现高级语言的优势。

Keil C51单片机软件开发系统的整体结构

C51工具包的整体结构，uVision与Ishell分别是C51 for Windows和for Dos的集成开发环境(IDE)，可以完成编辑、编译、连接、调试、仿真等整个开发流程。开发人员可用IDE本身或其它编辑器编辑C或汇编源文件。然后分别

由C51及A51编译器编译生成目标文件(.OBJ)。目标文件可由LIB51创建生成库文件，也可以与库文件一起经L51连接定位生成绝对目标文件(.ABS)。ABS文件由OH51转换成标准的Hex文件，以供调试器dScope51或tScope51使用进行源

代码级调试，也可由仿真器使用直接对目标板进行调试，也可以直接写入程序存贮器如 EPROM 中。KEILC51 标准 C 编译器为 8051 微控制器的软件开发提供了 C 语言环境,同时保留了汇编代码高效,快速的特点。C51 编译器的功能不断增强,使你可以更加贴近 CPU 本身,及其它的衍生产品。C51 已被完全集成到 uVision2 的集成开发环境中,这个集成开发环境包含：编译器,汇编器,实时操作系统,项目管理器,调试器。uVision4 IDE 可为它们提供单一而灵活的开发环境。

第二部分 uVision4 集成开发环境

一. 项目管理

工程(project)是由源文件、开发工具选项以及编程说明三部分组成的。

一个单一的 uVision4 工程能够产生一个或多个目标程序。产生目标程序的源文件构成“组”。开发工具选项可以对应目标，组或单个文件。

uVision4 包含一个器件数据库(device database)，可以自动设置汇编器、编译器、连接定位器及调试器选项，来满足用户充分利用特定 微控制器的要求。此数据库包含：片上存储器和外围设备的信息，扩展数据指针(extra data pointer)或者加速器(math accelerator)的特 性。

uVision4 可以为片外存储器产生必要的连接选项：确定起始地址和规模。

第三部分编辑器和调试器

一、源代码编辑器

uVision4 编辑器包含了所有用户熟悉的特性。彩色语法显像和文件辩识都对 C 源代码进行和优化。可以在编辑器内调试程序，它能提供一种自然的调试环境，使你更快速地检查和修改程序。

二、断点

uVision4 允许用户在编辑时设置程序断点（甚至在源代码未经编译和汇编之前)。用户启动 V2 调试器之后，断点即被激活。断点可设置为条件表达式，变量或存储器访问，断点被触发后，调试器命令或调试功能即可执行。

在属性框(attributes column)中可以快速浏览断点设置情况和源程序行的位置。代码覆盖率信息可以让你区分程序中已执行和未执行的部分。

三、调试函数语言

uVision4 中，你可以编写或使用类似 C 的数语言进行调试。

1.内部函数：如 printf, memset, rand 及其它功能的函数。

2.信号函数:模拟产生 CPU 的模拟信号和脉冲信号(simulate analog and digital inputs to CPU)。

3.用户函数：扩展指令范围，合并重复动作。

3 系统硬件电路的设计

3.1 单片机最小系统

单片机最小系统如图 3-1

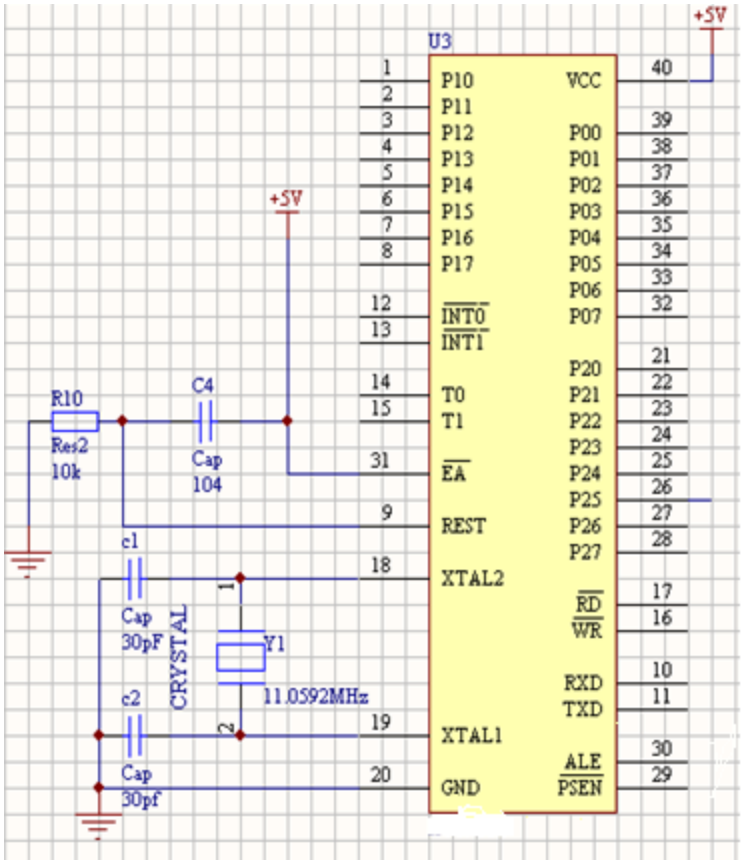


图 3-1　单片机最小系统

对图 3-1 说明如下：

（1） 单片机晶振电路

单片机外围的晶振电路是通过单片机的第 18（XTAL1）,19（XTAL2）引脚接入，XTAL1：振荡器反相放大器和内部时钟发生电路的输入端。XTAL2：振荡器反相放大器的输出端。

对于 STC11CF01 一般的晶振频率可以在 12MHz—35MHz 之间选择，这时电容 C 可以对应的选择 10pF—30pF。对于本设计的电容 C 用 30pF，晶振选用 35MHz。晶振电路解法图 3-1，一条引脚接在 XTAL1，另一条接在 XTAL2。

（2） 单片机的复位电路

RST：复位输入。晶振工作时，RST脚将持续2个机器周期高电平将使单片机复位。看门狗计时完成后，RST脚输出96个晶振周期的高电平。特殊寄存器AUXR(地址8EH)上的DISRTO位可以使此功能无效。DISRTO默认状态下，复位高电平有效。为了防止程序执行过程中失步或运行紊乱，此处我们采用了上电复位及手动复位电路，

（3）$\overline{EA}$/VPP：访问外部程序存储器控制信号。为使能从0000H到FFFFH的外部程序存储器读取指令，$\overline{EA}$必须接GND。为了执行内部程序指令，$\overline{EA}$应该接VCC。在flash编程期间，$\overline{EA}$也接收12伏VPP电压。

（4）ALE/$\overline{PROG}$：地址锁存控制信号（ALE）是访问外部程序存储器时，锁存低 8 位地址的输出脉冲。在 flash 编程时，此引脚（$\overline{PROG}$）也用作编程输入脉冲。在一般情况下，ALE 以晶振六分之一的固定频率输出脉冲，可以用来作为外部定时器或时钟使用。然而，特别强调，在每次访问外部数据存储器时，ALE 脉冲将会跳过。如果需要，通过将地址为 8EH 的 SFR 的第 0 位置 “1”，ALE 操作将无效。这一位置 “1”，ALE 仅在执行 MOVX 或 MOVC 指令时有效。否则，ALE 将被微弱拉高。这个 ALE 使能标志位（地址为 8EH 的 SFR 的第 0 位）的设置对微控制器处于外部执行模式下无效。

3.2 单片机的接口电路

（1）ZLG7289 芯片引脚介绍

1-2 VDD 正电源

3 5 NC 悬空

4 VSS 接地

6 CS 片选输入端此引脚为低电平时可向芯片发送指令及读取键盘数据

7 CLK 同步时钟输入端向芯片发送数据及读取键盘数据时此引脚电平上升沿表示数据有效

8 DATA 串行数据输入/输出端当芯片接收指令时此引脚为输入端当读取键盘数据时此引脚在读指令最后一个时钟的下降沿变为输出端

9 KEY 按键有效输出端平时为高电平当检测到有效按键时此引脚变为低电平

10-16 SG-SA 段g—段a 驱动输出

17 DP 小数点驱动输出

18-25 DIG0-DIG7数字0 数字7 驱动输出

26 OSC2 振荡器输出端

27 OSC1 振荡器输入端

28 RESET

zlg7289A 的指令结构有三种类型1.不带数据的纯指令指令的宽度为8 个BIT 即微处理器需发送8 个CLK 脉冲2.带有数据的指令宽度为16 个BIT 即微处理器需发送16 个CLK 脉冲3.读取键盘数据指令宽度为16 个BIT 前8 个为微处理器发送到zlg7289A 的指令后8 个BIT 为zlg7289A 返回的键盘代码执行此指令时zlg7289A 的DATA 端在第9 个CLK 脉冲的上升沿变为输出状态并与第16 个脉冲的下降沿恢复为输入状态等待接收下一个指令

串行接口的时序如下图：

1．纯指令：

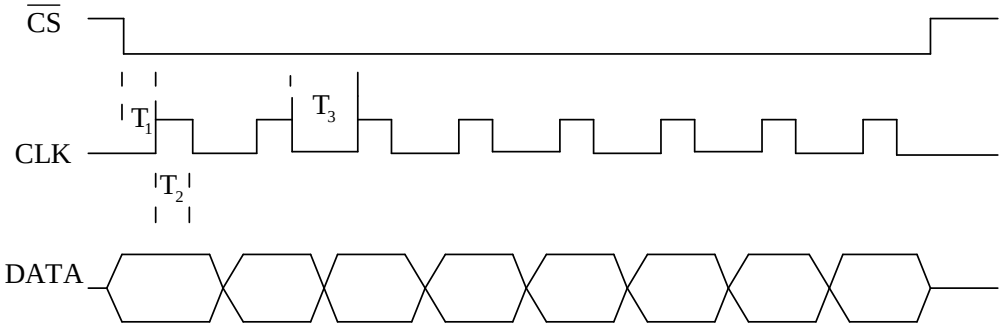


图3-2　纯指令时序图

2．带数据指令：

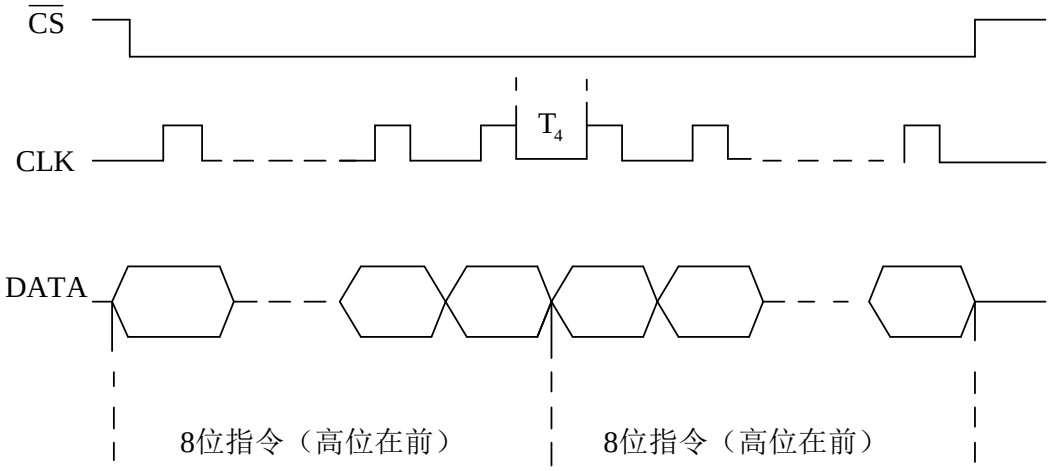


图3-2　带数据指令时序图

3．读键盘指令：

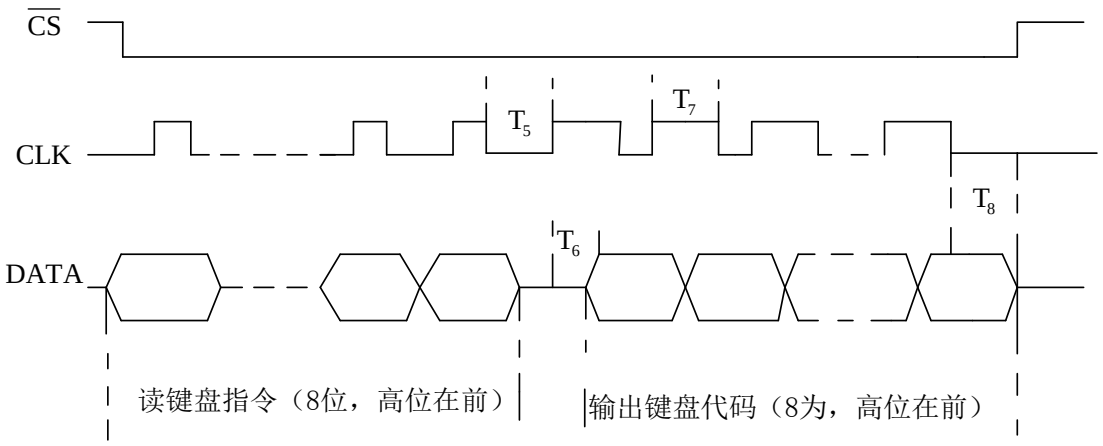


图3-3　读键盘指令时序图

$T_1=50\mu S, T_2=T_3, T_5=25\mu S, T_6=8\mu S,$

（2）ZLG7289 的接口电路图如图 3-4 所示。

zlg7289A应连接共阴式数码管应用中无需用到的数码管和键盘可以不连接省去数码管和对数码管设置消隐属性均不会影响键盘的使用如果不用键盘则典型电路中连接到键盘的8只10K电阻和8只100K下拉电阻均可以省去如果使用了键盘则电路中的8只10K电阻和8只100K 下拉电阻均不得省略除非不接数码管否则串入DP 及SA-SG 连线的8 只电阻均不能省去实际应用中8只下拉电阻和8只键盘连接位选线DIG0-DIG7的8只电阻位选电阻应遵从一定的比例关系下拉电阻应大于位选电阻的5倍而小于其50倍典型值为10倍下拉电阻的取值范围是10K-100K位选电阻的取值范围是1K-10K在不影响显示的前提下下拉电阻应尽可能的取较小的值这样可以提高键盘部分的抗干扰能力因为采用循环扫描的工作方式如果采用普通的数码管亮度有可能不够采用高亮或超高亮的型号可以解决这个问题数码管的尺寸也不宜选的过大一般字符高度不超过1 英寸如使用大型的数码管应使用适当的驱动电路zlg7289A 需要一外接晶体振荡电路供系统工作其典型值分别为F=16MHz C=15P如果芯片无法正常工作请首先检查此振荡电路在印刷电路板布线时所有元件尤其是振荡电路的元件应尽量靠近zlg7289A 并尽量使电路联线最短zlg7289A 的RESET 复位端在一般应用情况下可以直接和VCC 相连在需要较高可靠性的情况下可以连接一外部复位电路或直接由MCU 控制在上电或RESET 端由低电平变为高电平zlg7289A 大约要经过18-25MS 的时间才会进入正常工作状态上电后所有的显示均为空所有显示位的显示属性均为显示及不闪烁当有键按下时KEY 引脚输出低电平此时如果接收到读键盘指令。

zlg7289A 将输出所按下键的代码键盘代码的定义中代码以10进制表示如果在没有按键的情况下收到读键盘指令zlg7289A 将输出0FFH 255程序中尽可能地减少CPU 对zlg7289A 的访问次数可以使得程序更有效率因为芯片直接驱动LED 数码管显示电流较大且为动态扫描方式故如果该部分电路电源连线较细较长可能会引入较大的电源噪声干扰在电源的正负极并入一47U 到220U的电容可以提高电路抗干扰的能力注意如果有2 个键同时按下zlg7289A 将只能给出其中一个

键的代码因此zlg7289A不适于应用在需要2 个或2 个以上键同时按下的场合。

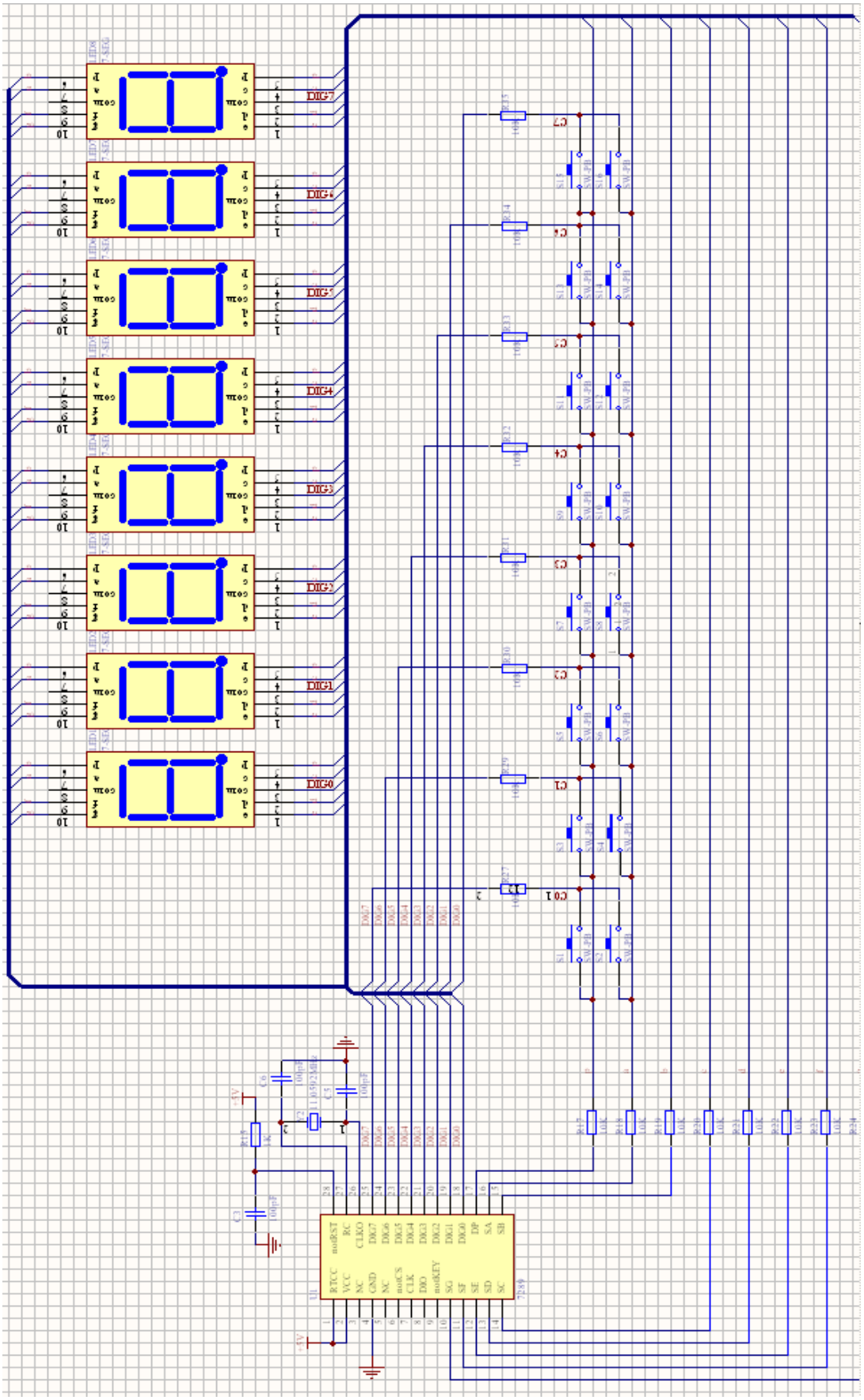


图 3-4　单片机与 ZLG7289 连接电路

3.3 幅度控制模块

3.3.1 单片机与 DAC0832 的接口

由于 D/A 转换器与单片机连接时，单片机是靠指令输出数字量供数模转换之用，而指令送出的数据在数据总线上的时间是短暂的，所以在 DAC 和单片机之间，需要有数据寄存器来保持单片机计算机输出的数据，供 DAC 转换使用。目前生产的 DAC 芯片分为两类，一类芯片内部设置有数据寄存器，不需要外加电路就可以直接与微型计算机接口。另一类芯片内部没有数据寄存器，输出信号随数据输入线的状态变化而变化，因此不能直接与微型计算机接口，必须通过并行接口与微型计算机接口。DAC0832 是具有 20 条引线的双列直插式 CMOS 器件，它内部具有两级数据寄存器，完成 8 位电流 D/A 转换。故不需要外加电路。因此单片机与 DAC0832 连接方式如图 3-5

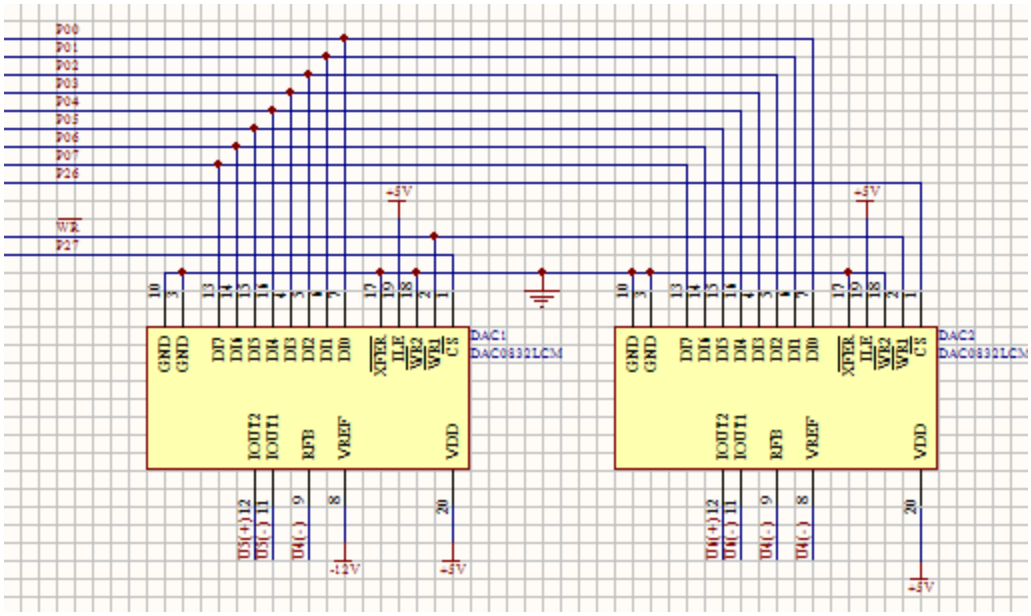


图 3-5　　D/A 接线图

3.3. 2DAC0832 与运放的连接

1. 在本设计中的运放的连接方式第一部分如图 3-6

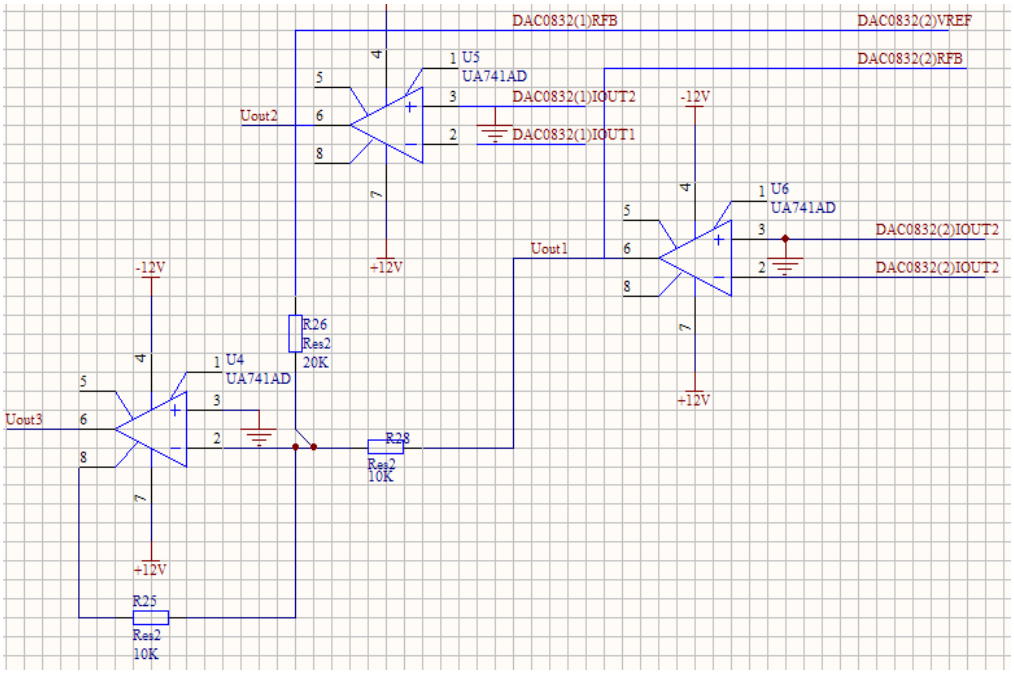


图 3-6　三运放连接图

上图 3-5 中的三个运放实现的功能分为：U6 作为一个反向比例器，U5 也是一个反向比例器，然而 U4 作为一个加法器，其理论的计算分析见第二章的第四节电路计算分析。实际上 Uout3 的输出结果为 Uout1 与 Uout2 的和。

2. 本设计中的运放的连接的第二部分如图 3-7

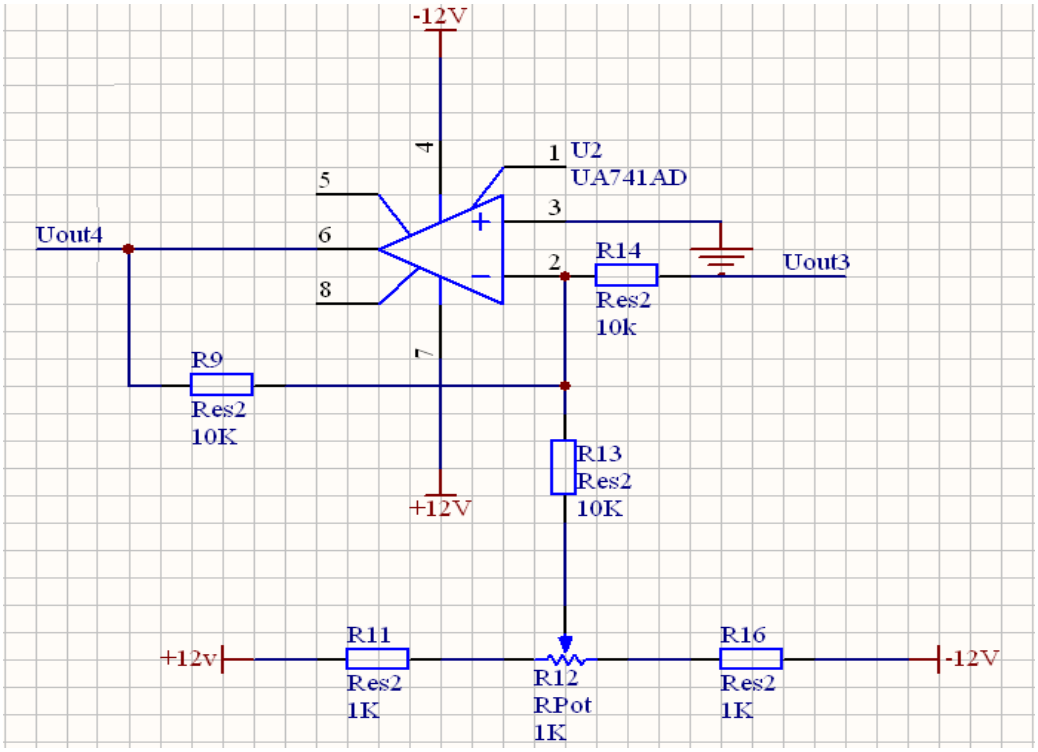


图 3-7　直流偏移调节运放连接图

该电路由 102 电位器串接 2 个 1kΩ 电阻实现调节直流偏移，电位器触头在最右端和最左端时，电位器输出的电压分别为-5v 和+5v，电位器的电压与 U_0 通过一

个加法器后，实现直流偏移的调节。

2．电路性能指标分析

用于调压的DAC0832的参考电压是12V，所以，峰峰值可以调节到的最大值为12V，由于运放的电源均为±12V，故均未达到饱和。

通过1 kΩ电位器与两个1 kΩ的电阻进行直流偏移的调节。当电位器的滑动触头分别位于最右端与最左端时，输出电压分别为-5v和+5v，电位器的电压与输出的电压通过一个加法器实现直流偏移的调节。

由于DAC0832存在的非线性，输出信号的幅值存在一定的误差。

由上述计算可知，该电路产生波形的峰峰值和直流偏移调节的范围达到并超过了题目要求的范围。

4 系统软件设计

4.1 系统软件设计方案

51 单片机系列的编程语言常用的有两种，一种是汇编语言，一种是 C 语言。

汇编语言，是一种功能很强的程序设计语言，也是利用计算机所有硬件特性并能直接控制硬件的语言。汇编语言直接同计算机的底层软件甚至硬件进行交互，它具有如下一些优点：(1)能够直接访问与硬件相关的存储器或 I/O 端口；(2)能够不受编译器的限制，对生成的二进制代码进行完全的控制；(3)能够对关键代码进行更准确的控制，避免因线程共同访问或者硬件设备共享引起的死锁；(4)能够根据特定的应用对代码做最佳的优化，提高运行速度；(5)能够最大限度地发挥硬件的功能。同时还应该认识到，汇编语言是一种层次非常低的语言，它仅仅高于直接手工编写二进制的机器指令码，因此不可避免地存在一些缺点：(1)编写的代码非常难懂，不好维护；(2)很容易产生 bug，难于调试；(3)只能针对特定的体系结构和处理器进行优化；(4)开发效率很低，时间长且单调。

C 语言，是一种计算机程序设计语言。它既具有高级语言的特点，又具有汇编语言的特点。它可以作为工作系统设计语言，编写系统应用程序，也可以作为应用程序设计语言，编写不依赖计算机硬件的应用程序。因此，它的应用范围广泛，不仅仅是在软件开发上，而且各类科研都需要用到 C 语言，具体应用比如单片机以及嵌入式系统开发。它具有如下优点：(1)简洁紧凑、灵活方便；(2)运算符丰富；(3)数据结构丰富；(4)C 是结构式语言；(5)C 语法限制不太严格，程序设计自由度大；(6)C 语言允许直接访问物理地址，可以直接对硬件进行操作；(7)生成目标代码质量高，程序执行效率高；(8)C 语言适用范围大，可移植性好。

汇编语言的机器代码生成效率很高但可读性却并不强，复杂一点的程序就更是难读懂，而 C 语言在大多数情况下其机器代码生成效率和汇编语言相当，但可读性和可移植性却远远超过汇编语言，而且 C 语言还可以嵌入汇编来解决高时效性的代码编写问题。对于开发周期来说，中大型的软件编写用 C 语言的开发周期通常要小于汇编语言很多。综合以上 C 语言的优点，函数发生器系统的软件部分由 C 语言设计编程实现。

4.2 系统软件流程图

系统软件是由若干子程序构成，包括主程序、显示子程序、各模式子程序等等。主程序的流程图如图 4-1 所示

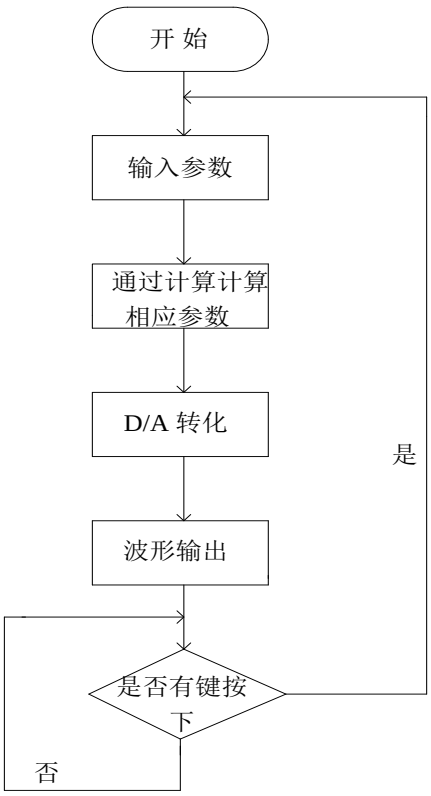


图 4-1 主程序流程图

2.键盘输入的流程图如图 4-2

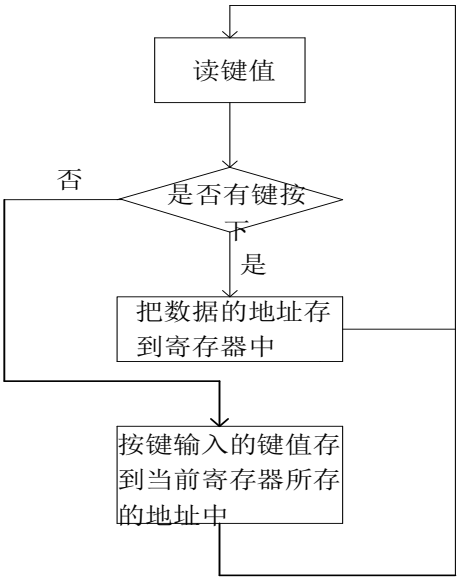


图 4-2 键盘输入流图

4.3 信号产生程序

本设计将各种波形的数据固定在单片机的程序存储器里，通过改变这些数据的输出速度来改变信号的频率，然后通过改变 D/A 转换器的参考点要来改变信号的峰峰值，电路较为简单，成本较低。

4.3.1 正弦波产生

方法一：首先在单片机的存储器中存入正弦波的点数，通过输出的两点间的延时来实现调频的功能。我们通过两个机器周期的语句作为调频的最小时间单位，通过循环次数来控制时间，假设波形点数为 n，输入频率为 f，每个频段的最小分辨率为 x。

方法二：直接输入计算式方法。

程序如下：

```
zhengxianbo()
{
    int  x,y;
    while(1)
    {    for(x=0;x<128;x++)
          {
         y=59*sin(2*3.1415926*x/128)+128;
         DA0832A    = y;
          }
     }
}
```

4.3.2 三角波产生

1．产生三角波的原理

设个自变量 i 让它不断地自加 1，直到加到 255 时，t=i ，对 t 进行不断地自减一直到减到 t=0,然后再不断地重复上述过程进而产生三角波。

2．程序流程图见图 4-3

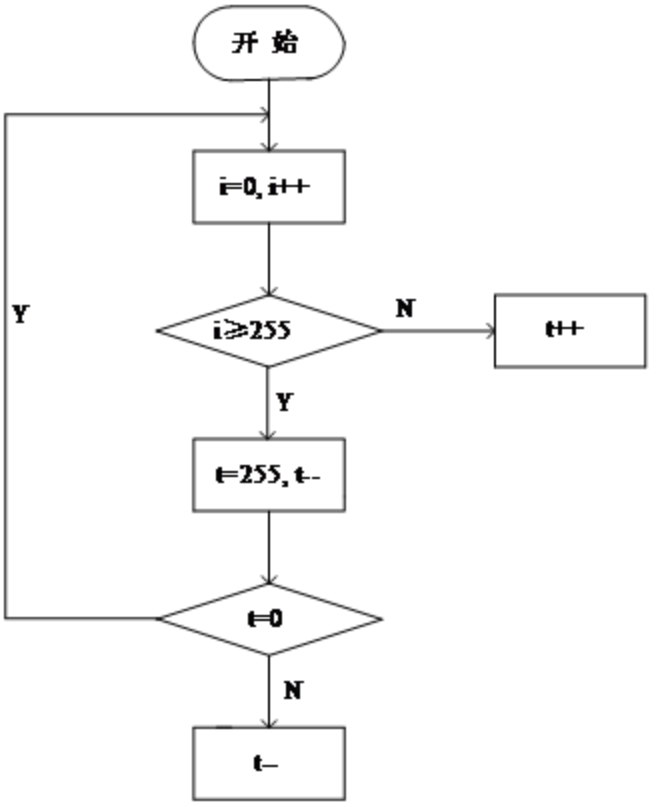


图 4-3　三角波流程图

```
void sanjiaobo()
{
    uchar i = 1,temp=0x00;
    while(1)
    {
    DA0832A=temp;
    temp+=i;
    if(temp==0xc0||temp==0x00)
    i=0-i;
    }
}
```

4.3.2 方波产生

1．产生方波的原理

设个自变量 i=0 使之延时一段时间，再另 i=255 时在延时与 i=0 相同的时间，然后在重复上述过程。假设延时为 T。

2.方波流程图见图 4-4

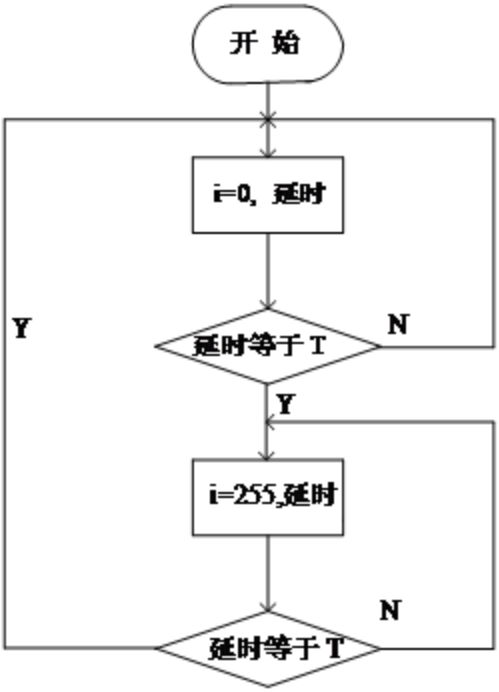


图 4-4　方波流程图

3．程序

```
fangbo()
{
    uchar i;
    CS=1;
    while(1)
    {
    if(i<0x7f)
    DA0832A=0x00;
    if(i>=0x7f)
    DA0832A=0xff;
    i++;
    if(i==0xff)
    i=0;
    }
}
```

4.3.4 锯齿波的产生

1．产生锯齿波的原理

锯齿波中的斜线用一个个小台阶来逼近，在一个周期内从最小值开始逐步递增，

当达到最大值后又回到最小值，如此循环，当台阶间隔很小时，波形基本上近似于直线。适当选择循环的时间，可以得到不同周期的锯齿波。锯齿波发生原理与方波类似，只是高低两个延时的常数不同，所以用延时法，来产生锯齿波，设个自变量 i 让它不断地自加 1，直到加到 255，DAC0832 可以又自动归 0，然后再不断地重复上述过程进而产生锯齿波。

3．程序

```
juchi()
{
    uchar temp=0x00;
    while(1)
    {
    DA0832A=temp++;

    if(temp==0xc0)
    temp=0;
    }
}
```

5 系统调试与测试

5.1 调试

简单系统硬件的调试通常采用载入简单的测试程序并运行，使用数字表或示波器观察；对有些硬件例如显示器、键盘等可直接编入程序观察程序执行状态。

1.上电复位后用示波器观察晶振或 ALE 是否有波形输出。如有表明单片机已激活。用示波器观察晶振波形如图 5-1 所示。

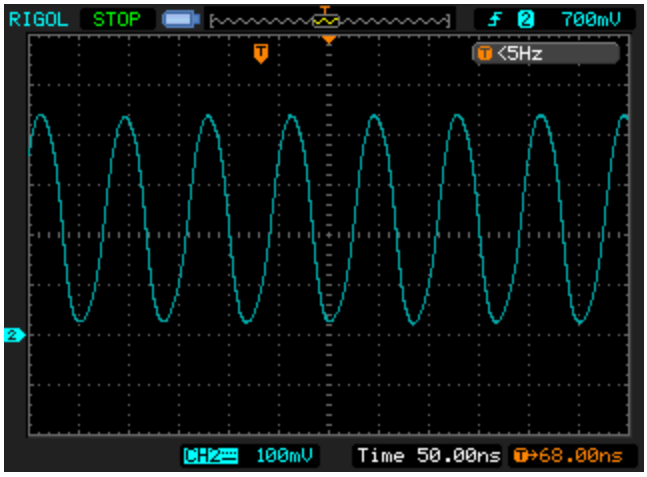


图 5-1　晶振波形

2.按键的调试

对于新给定的一个 ZLG7289 及键盘是，我们应当首先确定每个按键的键值，只有当我们确定键之后，才可以对我们所要设置的按键的功能通过真正做到使软硬件相结合在一起，实现我们的目标功能，真是我们可以利用系统的显示部分也即就是系统的 LED，我们通过假设法使出本系统的键值。一般对于设计心得系统我们都可以采用这种方法，因为现在是人性化社会每个设计都有自己的特色，在设计中我们的充分利用系统为我们所提供的一些显而易见的东西来完成我们所打算实现的功能。

对于十六个按键及功能介绍见表 5-1

表 5-1　按键功能表

KEY	作用
K1	0
K2	1
K3	2
K4	3
K5	4
K6	5
K7	6
K8	7
K9	8
K10	9
K11	小数点
K12	频率设置
K13	幅度设置
K14	波形切换
K15	LED 清屏
K16	确定键

通过仿真器对系统进行调试，使用调试软件为 KeilC51，软件版本为μVision4。系统上电运行后，第 1 次按下波形切换键 K14，再按下确认件 K16，从示波器上观察结果如图 5-3

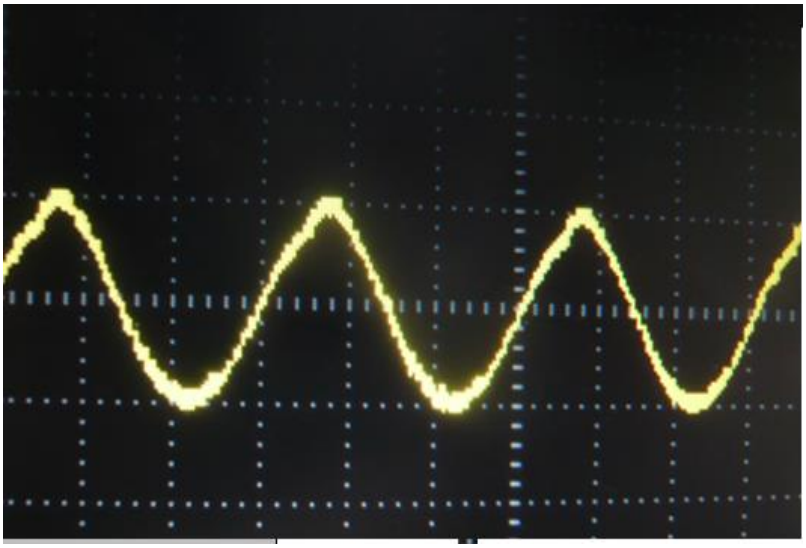


图 5-3　正弦波

第 2 次按下 K14 按确定 K16 后从示波器观察图形如图 5-4

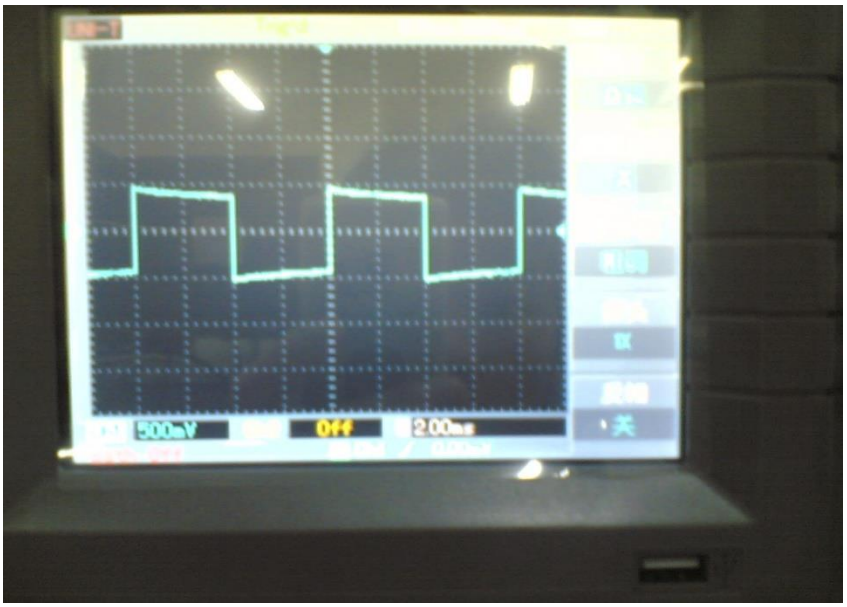


图 5-4　方波

第 3 次按下 K14 按确定 K16 后从示波器观察图形如图 5-5

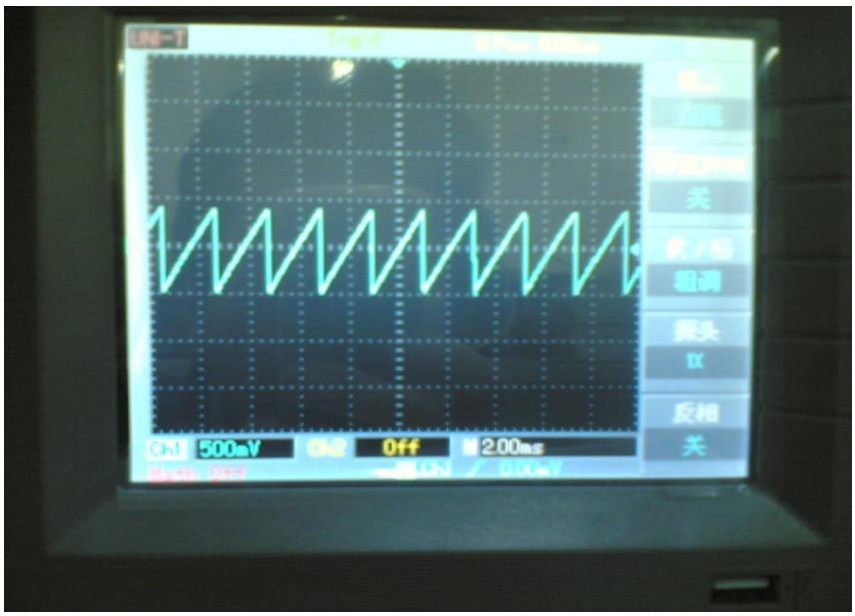


图 5-5 锯齿波

第 3 次按下 K14 按确定 K16 后从示波器观察图形如图 5-6

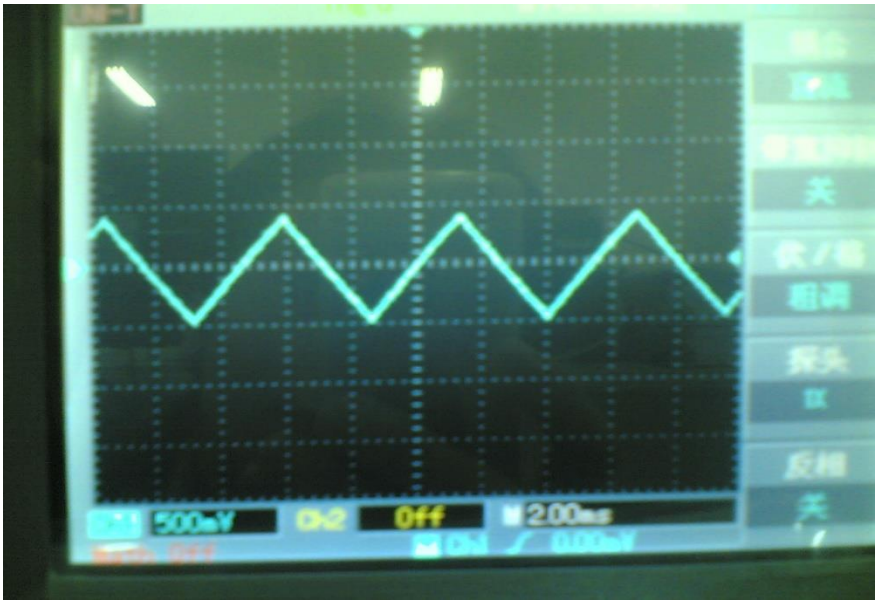


图 5-6 三角波

通过实验可以知道本系统中的 K14 用于几种波形的相互切换。

5.2 测试

1．测试的方法：

在测试本系统时采用的是将设置的数值与示波器所测试值进行对比，进而可以知道本系统的性能

2.通过按键，实现其按键所对应的功能，并观察测试结果，对设计进一步的进行

校正和对实现功能的可靠新的确认，并记录观察结果。

测试数据如下（以正弦波为例）

（1）电压峰值测试数据如表 5-1

表 5-1　电压峰值测试数据

设定电压（V）	示波器测试电压值（V）	相对误差值%
2.(	2.0	1.!
4.(	4.0	0.!
6.(	6.0	0.8
8.(	8.0	0.!

（2）频率测试数据如表 5-2

表 5-2　电压峰值测试数据

设定频率（Hz）	示波器测试电压值（Hz）	相对误差值%
2.0	2.0	0
8	7.94	0.625
10	9.56	0.44
20	19.93	0.35
40.0	39.1	0.45
80.0	79.64	0.45
100.0	104.8	0.48
200.0	201.2	0.6

所能测得正弦波的最高频率为 1MHz。

锯齿波和反锯齿波的最高频率为 1.01MHz

三角波测得的最高频率为 1.01Hz
方波测得的最高频率为 1.12MHz
3.数据分析
电压误差分析：
产生误差的原因有：DAC0832 的非线性、电压运算过程中四舍五入产生的误差、直流电源存在的误差、电阻及运放等器件存在的误差。
频率误差分析：
延时计算误差、示波器示数不稳定。
高频时存在较大误差的原因是，高频时调用延时程序的次数较少，因此计算调用次数时产生的误差较大。
频率存在上限的原因是程序运行的耗时以及 DAC0832 在进行 DA 转换时损耗的时间。
综上所述，由上述数据可以看出，函数信号发生器峰峰值和频率的误差还是很低的，精度较高。

6 结论与展望

6.1 结论

经过几个月的努力，终于完成了函数发生器的设计，总结几个月来来的工作，主要有以下几个方面：

（1）通过查阅大量资料使自己对函数发生器的研究现状、原理、工作方式等的基本概念及技术发展有了更好的理解。

（2）针对设计的任务和要求，确定了函数发生器的硬件和软件设计方案。

（3）本系统使用了单片机作为中央控制器，直接由软件产生波形信号的输出，可通过修改软件，还可以输出其它任意波形。硬件简单,可以把电源和发生器分开来制作，减少电源对发生器的干拢，这样使发生器输出的波形更加稳定。

（4）论述了函数发生器的调试方法和测试结果，基本完成了设计的要求。

（5）用 protel 99SE 设计并制作了函数发生器的印制电路板 PCB。

6.2 展望

由于时间和条件的限制，本系统也存在一些不足之处，可在今后的工作中改进。

（1）本系统的核心控制芯片单片机，由于单片机本身的晶振频率为 35MHz,因此得频率调节范围仅在 1～1M Hz，仅适用于中高频信号源的电路当中，对于超过 1M HZ 就无法满足了。为了使频率范围能够扩大，在以后的设计中，应采用更高晶振频率的控制器件如 DSP 等。

（2）设想在波形的输出口增加 A / D 模块电路对输出信号进行反馈，单片机对反馈信号进行误差的较验、调整再输出，在没有达到所要求的精度的波形输出时，再一次进行反馈、较验、调整直到符合所需要求的波形输出。或者我们可以利用更改精度的 A/D，例如 12 位的 A/D，MAX197。来提高输出波形的质量。

（3）直流偏移的调节目前是通过电位器实现的，并不能进行数字控制，可以再增加一片 DAC0832 产生直流电压，与产生的波形累加，可以实现数字调节

致 谢

在本次系统的研究和设计过程中，我得到了老师和同学们的热情帮助。在此，对他们表示衷心的感谢。

首先，要对我的指导老师马老师表示衷心的感谢。从方案的选取、审题、查找资料，到系统软硬件的各部分设计工作，到最后论文的书写和完成，老师在我的整个毕业设计工作中给了我很大的帮助和支持。老师的谆谆教导，使我受益匪浅。

其次，要对大学三年以来所有给我授课的老师们表示感谢。是他们教会了我大学应该掌握的知识和技能，给我打下了坚实的理论基础。只有运用三年学习的基础知识和经验的积累，才能使我能够顺利的完成本次毕业设计工作。

最后，要感谢我们班的众多同学，本次设计能够圆满完成，和各位同学的帮助是息息相关的。在本次设计中，我遇到了无数困难，在需要帮助的时候，各位同学给了我无私的帮助，助我度过了一个又一个的难题。

附 录

附录一 系统软件部分源程序

```
#include<reg52.h>
#include<absacc.h>
#include<math.h>

#define  pi 3.1415926
//#define    DEBUGZLG7289      1
//#define        DEBUGTIME0       1
sbit CS=P1^0;
sbit CLK=P1^1;
sbit DIO=P1^2;
sbit KEY=P1^3;

//#define  uchar unsigned  char
typedef unsigned  char uchar;
#define  DAC1    XBYTE[0x7fff]
#define  DAC2    XBYTE[0xbfff]

void key_function(void);
void key_number(void);

uchar  rebuf,sebuf;
unsigned  char amp=0x80;
uchar  number;          //按键数码设置
uchar  dis_bit=0;  //按键显示位数设置

bdata uchar com_data;

sbit mos_bit=com_data^7;
sbit low_bit=com_data^0;
//uchar   th=0xff;
//uchar   tl=0xff;
uchar  valid=0;                    //按键确认标志
uchar  fun=1;                  //函数类型声明

uchar  cnt=0;                  //a=0,b=0,c=0,d=0;        //函数计数
uchar  tl=0x00,th=0xf0;         //计数器计数初值              //fffd 计数值太快导致，
按键 while 循环无法正常工作
//uchar  tl=0x13,th=0x1c;
```

```
uchar c_max=35;
uchar                                                                    code
sint[36]={128,150,171,191,209,225,237,247,253,254,253,247,237,225,209,191,171,1
50,128,105,
84,64,46,30,18,8,2,1,2,8,18,30,46,64,84,105};


void delay( uchar n)              //8us*n
{
    uchar i;
    for(i=0;i<n;i++);
}

void receive(void)
{
    uchar i;
    //CS=0;
    CLK=1;
    delay(6);
    for(i=0;i<8;i++)
    {
        com_data=com_data<<1;
        low_bit=DIO;
        CLK=1;
        delay(1);
        CLK=0;
        delay(1);
    }
    rebuf=com_data;
    DIO=1;
    CS=1;
}

void send( uchar sebuf)
{
    uchar i;
    com_data=sebuf;
    CLK=0;
    CS=0;
    DIO=0;
    delay(6);
    for(i=0;i<8;i++)
    {
        delay(1);
```

```
        DIO=mos_bit;
        CLK=1;
        delay(1);
        com_data=com_data<<1;
        CLK=0;

    }
    DIO=0;
    //CS=1;             //          CS=1 时字节输入结束用于 7289 正常显示

}



void reset()
{
    KEY=1;
    DIO=1;
    delay(6);
    send(0xa4);
    CS=1;
}

void time0_int(void) interrupt 1 //中断服务程序
{
    //TR0=0;
    if(fun==0)              //锯齿波(频率除 2）
    {

        DAC1=amp;

        DAC2=(uchar)((float)(cnt)*0xff/c_max);
        if(cnt<c_max)     cnt++;
        else              cnt=0;

    }
    else if(fun==1)
    {
        DAC1=amp;
        //DAC2=sint[cnt]; //正弦波
        DAC2=sint[cnt]; //正弦波
        //DAC2=128+127*sin(2*pi*cnt/256);
        if(cnt<c_max)     cnt++;
        else              cnt=0;
```

```
        //cnt++;

    }
    else if(fun==2) //三角波
    {
        DAC1=amp;
        //if(cnt<128) DAC2=cnt;
        //else        DAC2=255-cnt;
        if(cnt<(uchar)((c_max-1)/2.0)+1)   DAC2=(uchar)((float)(cnt)*0xff/c_max);
        else
    DAC2=255-(uchar)((float)(cnt)*0xff/c_max);
        if(cnt<c_max)     cnt++;
        else              cnt=0;
    }
    else if(fun==3) // 方波
    {
        if(cnt<c_max)     cnt++;
        else              cnt=0;
        DAC1=amp;
        if(cnt<(uchar)((c_max-1)/2.0+1))   DAC2=0x00;
        //if(cnt<128)       DAC2=0x00;
        else              DAC2=0xff;

    }
    else if(fun==4)               //直流
    {
        DAC1=amp;
        DAC2=0xff;
    }
    TH0=th;
    TL0=tl;
    //TR0=1;
}

void down0_show(uchar  dis_bit)
{
    send(0xa1);
    send(0x80|dis_bit);
    delay(6);
    send(number);
    delay(6);
    CS=1;
}
```

```
void down1_show(uchar  dis_bit)
{
      send(0xa1);
      send(0xc8|dis_bit);
      delay(6);
      send(number);
      delay(6);
      CS=1;
}

void set_point(dis_bit)
{
      send(0x80|dis_bit);
      delay(6);
      number=(0x80|number);
      send(number);
      delay(6);
      CS=1;
}

void set_freq(void)
{
      float  freq=0;
      unsigned  int temp;
      unsigned  int n;
      uchar dot=0;

      while(!KEY);
      KEY=1;
      //send(0x98);
      //delay(6);
      //send(0x07);
      //delay(6);
      dis_bit=3;
      number=0x0f;
      down1_show(dis_bit);
      valid=0;
      while(valid==0)
      {
            while(KEY);
            key_number();
            if(valid==0)
            {
                  if(number>=0x80)      {     dot=1;}
```

```
            else
            {
                if(dot==1)
                {
                    //number=(number&0x0f);
                    freq=freq+number/10.0;
                    dot=0;
                }
                else freq=freq*10+number;
            }
        }

        while(!KEY);
        KEY=1;

    }
    if(freq>100)  freq=freq*1.2;
    if(freq>150)  freq=freq*1.2;
    if(freq>200)  freq=freq*1.1;
    if(freq>230)  freq=freq*1.1;
    n=(unsigned  int)(4266.666/freq*5);

    temp=65536-n;
    th=(temp>>8);
    tl=temp;

}
void set_amp(void)
{

    float temp;
    uchar dot=0;
    while(!KEY);
    KEY=1;
    //send(0x98);
    //delay(6);
    //send(0xf8);
    //delay(6);
    dis_bit=0;
    number=0x0a;
    down1_show(dis_bit);
    valid=0;
    while(valid==0)
    {
```

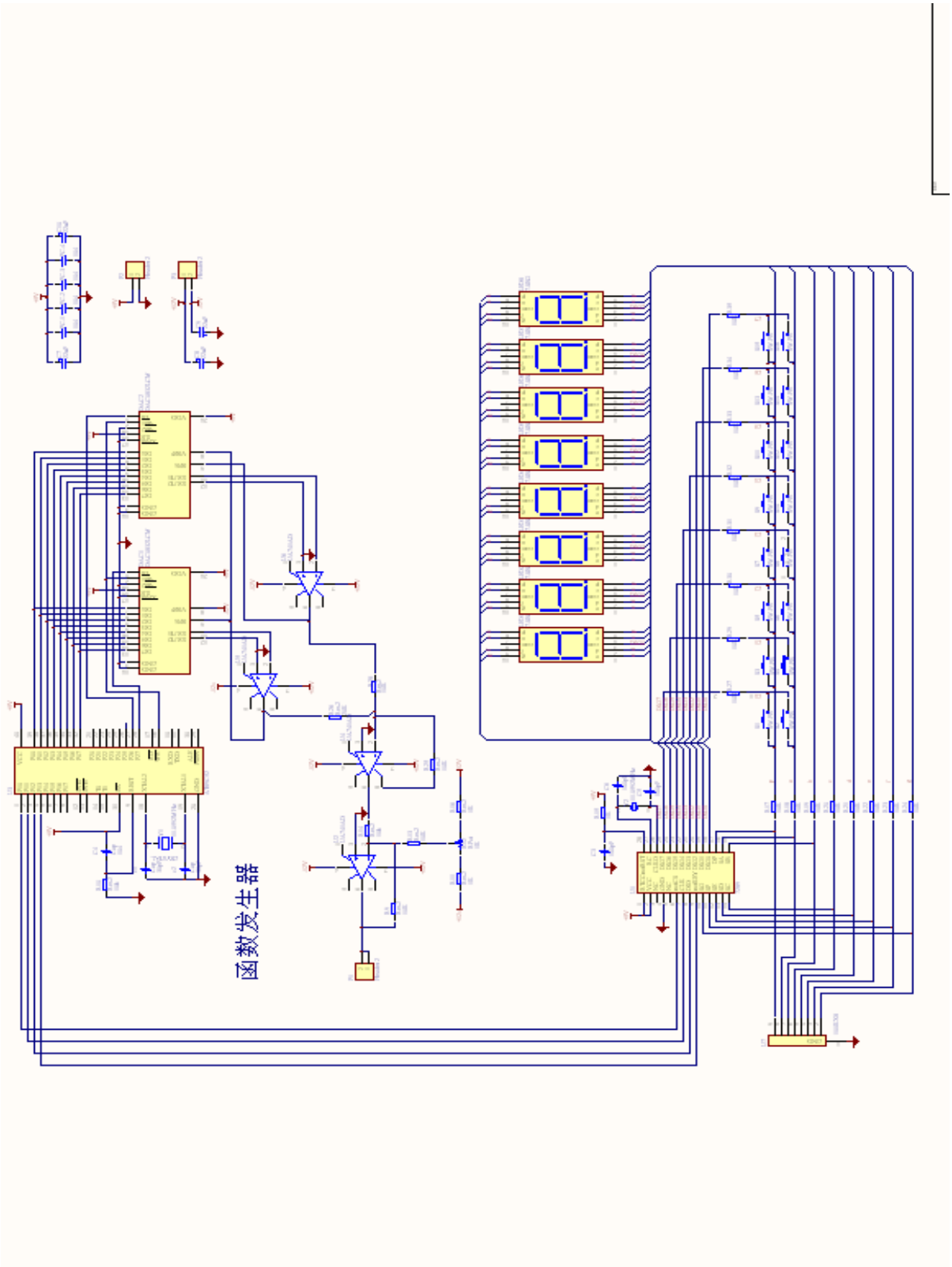
```
            while(KEY);
            key_number();
            if(valid==0)
            {
                if(number>=0x80)    {    dot=1;}
                else
                {
                    if(dot==1)
                    {
                        //number=(number&0x0f);
                        temp=temp+number/10.0;
                        dot=0;
                    }
                    else temp=temp*10+number;
                }
            }
            while(!KEY);
            KEY=1;
        }
        amp=temp/12.0*0xff;
}

void key_number(void)
{
        send(0x15);
        delay(6);
        receive();
        delay(6);
        switch(rebuf)
            {
                case 0x06:      {    number=0;    down0_show(dis_bit);      break;}
                case 0x0e:      {    number=1;    down0_show(dis_bit);      break;}
                case 0x16:      {    number=2;    down0_show(dis_bit);      break;}
                case 0x1e:      {    number=3;    down0_show(dis_bit);      break;}
                case 0x26:      {    number=4;    down0_show(dis_bit);      break;}
                case 0x2e:      {    number=5;    down0_show(dis_bit);      break;}
                case 0x36:      {    number=6;    down0_show(dis_bit);      break;}
                case 0x3e:      {    number=7;    down0_show(dis_bit);      break;}
                case 0x07:      {    number=8;    down0_show(dis_bit);      break;}
                case 0x0f:      {    number=9;    down0_show(dis_bit);      break;}
                case 0x17:      {set_point(dis_bit);      break;}
                case 0x3f:      {valid=1;                 break;}
            }
}
```

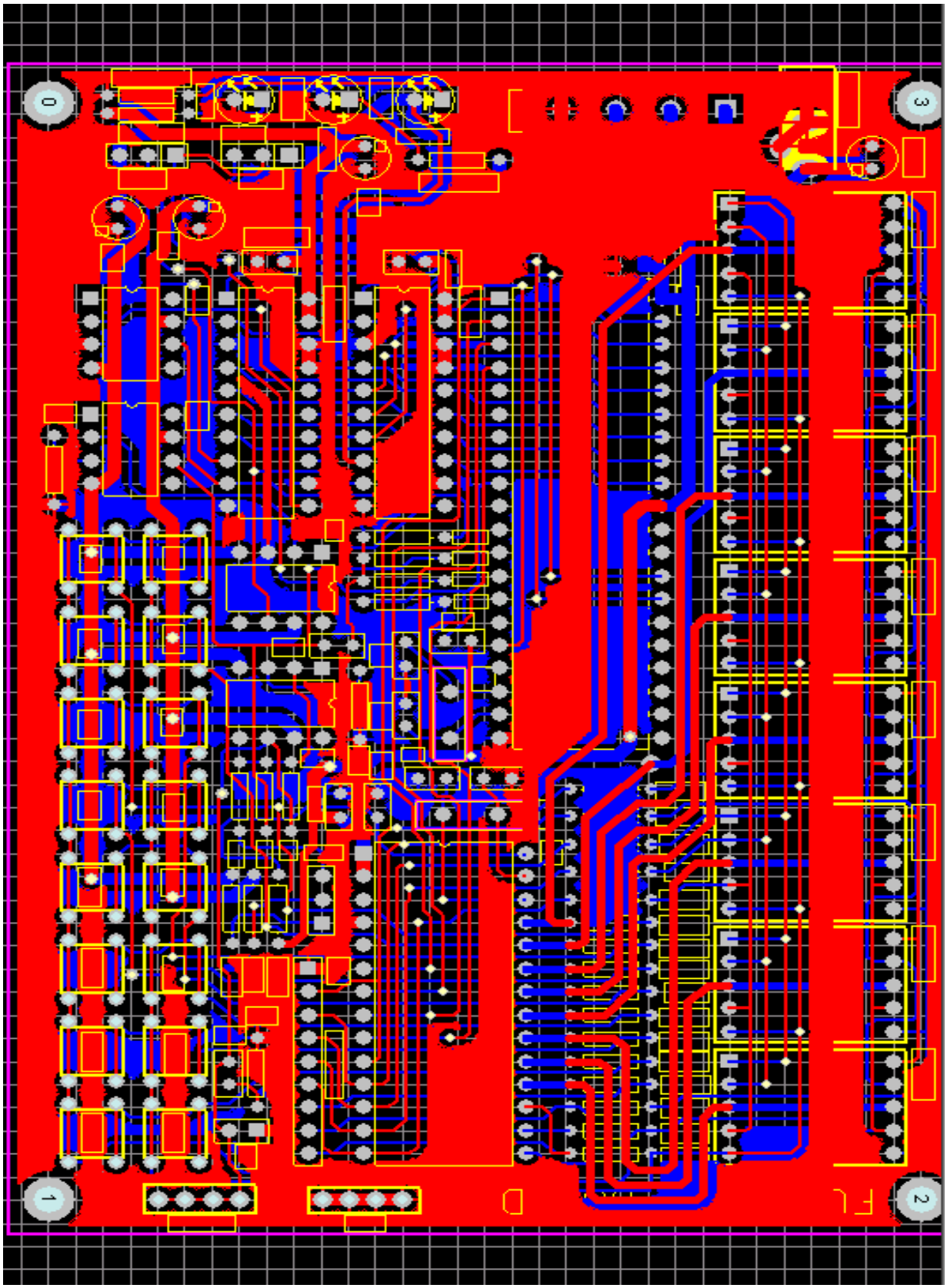
```
void key_function(void)
{
    send(0x15);
    delay(6);
    receive();
    delay(6);
    switch(rebuf)
        {
            case 0x06:      {   number=0;   down0_show(dis_bit);    break;}
            case 0x0e:      {   number=1;   down0_show(dis_bit);    break;}
            case 0x16:      {   number=2;   down0_show(dis_bit);    break;}
            case 0x1e:      {   number=3;   down0_show(dis_bit);    break;}
            case 0x26:      {   number=4;   down0_show(dis_bit);    break;}
            case 0x2e:      {   number=5;   down0_show(dis_bit);    break;}
            case 0x36:      {   number=6;   down0_show(dis_bit);    break;}
            case 0x3e:      {   number=7;   down0_show(dis_bit);    break;}
            case 0x07:      {   number=8;   down0_show(dis_bit);    break;}
            case 0x0f:      {   number=9;   down0_show(dis_bit);    break;}
            case 0x17:      {   set_point(dis_bit);         break;}
            case 0x1f:      {   set_freq();                 break;}
            case 0x27:      {   set_amp();                  break;}
            case 0x2f:      {   if(fun<4)   fun++;
                                else        fun=0;      break;}
            case 0x37:      {   reset();                break;}
            case 0x3f:      {   valid=1;                break;}
            default:        break;
            }
}
void main()
{
    //CS=1;       CS=1 时 7289 正常显示
    //send(0xbf);       //测试 7289 是否正常工作
    /*
    uchar i;
    for(i=0;i<64;i++)
    {   t2sin[i]=32+31*sin(2*pi*i/64);}
    for(i=0;i<32;i++)
    {   t3sin[i]=16+15*sin(2*pi*i/32);}
    */
    reset();
    //c_max=143;
//#if       DEBUGTIME0
    TMOD=0x01;
```

```
    TH0=th;
    TL0=tl;
    EA=1;
    ET0=1;
    TR0=1;
    //while(1);
    while(1)
    {
        while(KEY);
        key_function();
        while(!KEY);
        KEY=1;
    }
//#endif
    //dis_bit=0;
    //while(1);
//#if  DEBUGZLG7289
    //delay(1);
    //
//#endif
}
```

附录二 系统原理图



附录三 系统 PCB 图



参 考 文 献

[1] 王建校等，《51 系列单片机原理及 C 语言程序设计》. 科学出版社，2002 年 4 月.
[2] 王建校等，《电子系统设计与实践》. 高等教育出版社，2008 年 5 月。
[3] 张鑫，华臻，陈书谦.《单片机原理及应用》. 电子工业出版社，2005 年 8 月.
[4] 胡汉才.《单片机原理及其接口技术》. 北京：清华大学出版社，1996 年 7 月.
[5] 申忠如.《MCS-51 单片机原理及其系统设计》. 西安交通大学出版社，2008 年 3 月.
[6] 张克农等，《数字电子技术基础》，高等教育出版社，2005 年
[7] 刘君华，《现代检测技术与测试系统设计》，西安交通大学出版社，1999 年
[8] 郭戍生、古天祥、陆玉新、张世箕，《电子仪器原理》，国防工业出版社，1989 年
[9] 赵新民、王祁，《智能仪器设计基础》，哈尔滨工业大学出版社，1999 年
[10] 夏路易. 电路原理图与电路板设计教程 Protel99se [M]. 北京希望电子出版社，2002
[11] Huelsman,L.P.Badic 《Circuit Theory (2nd edition).Prentice-Hall, 》
[12] uelsman,L.P.Badic 《Circuit Theory (2nd edition).Prentice-Hall, 》