華中科技大學

电子台签技术手册

华中科技大学 Dian 团队

2008 级种子班

李海涛 陈曦骏 龚小聪

2010 年 11 月 11 日

目录

图目录

表目录

1 概述

当前，信息化建设在各地蓬勃发展，作为信息发布的终端显示设备，LED 显示屏已经广泛应用于工作和生活的各个方面。电子台签就是 LED 电子显示屏的简单应用。

本电子台签的设计是针对种子班的《微机原理》的课程设计。该课程设计重在理解单片机内部的工作原理，即通过软件控制单片机内部的工作。课程设计环节除了检测、巩固、综合课程中学习的知识外，还为提供一个熟悉嵌入式系统设计开发流程的平台，使设计者熟悉简单的嵌入式开发流程，积累相关经验。

本电子台签由华中科技大学 2008 级种子班自主研发完成。实验板采用 Atmel 公司的 AT89S52 芯片作为主控芯片，显示部分采用了 16 块 8*8 的 LED 点阵屏，外加 7 个普通操作按键和 1 个模式切换按键，实现了以下功能：

- 具备 4 汉字显示功能。
- 具备时钟显示功能
- 可通过上位机软件更新显示内容。
- 可采用 220V 交流电直接供电。

2 整体设计

2.1设计思路

本电子台签通过上位机软件，在 PC 机上将要显示的汉字字模提取出来，并通过 USB 转串口发送给单片机，下载的字模存储在 EEPROM 中，然后显示在 LED 点阵屏上，整机通过 220 交流电直接供电。主要应用场景是桌面台签的显示，如会议嘉宾的姓名显示等。

我们采用的是 Ateml 公司的 AT89S52 单片机作为实验板的主控芯片。它有 32 个可编程 I/O 接口（4 个 8 位并行 I/O 接口），3 个 16 位定时器/计数器，6 个中断源、2 个优先级嵌套中断结构，1 个全双工 UART 串口以及 8KB 的程序存储器、256B 数据存储器，可以满足本产品的汉字显示需要，及其他按键扩展功能的需要。

2.2系统框图

我们的实验板以功能划分模块，相互独立地进行设计整合，共计有 7 个模块，我们的系统框图如下：

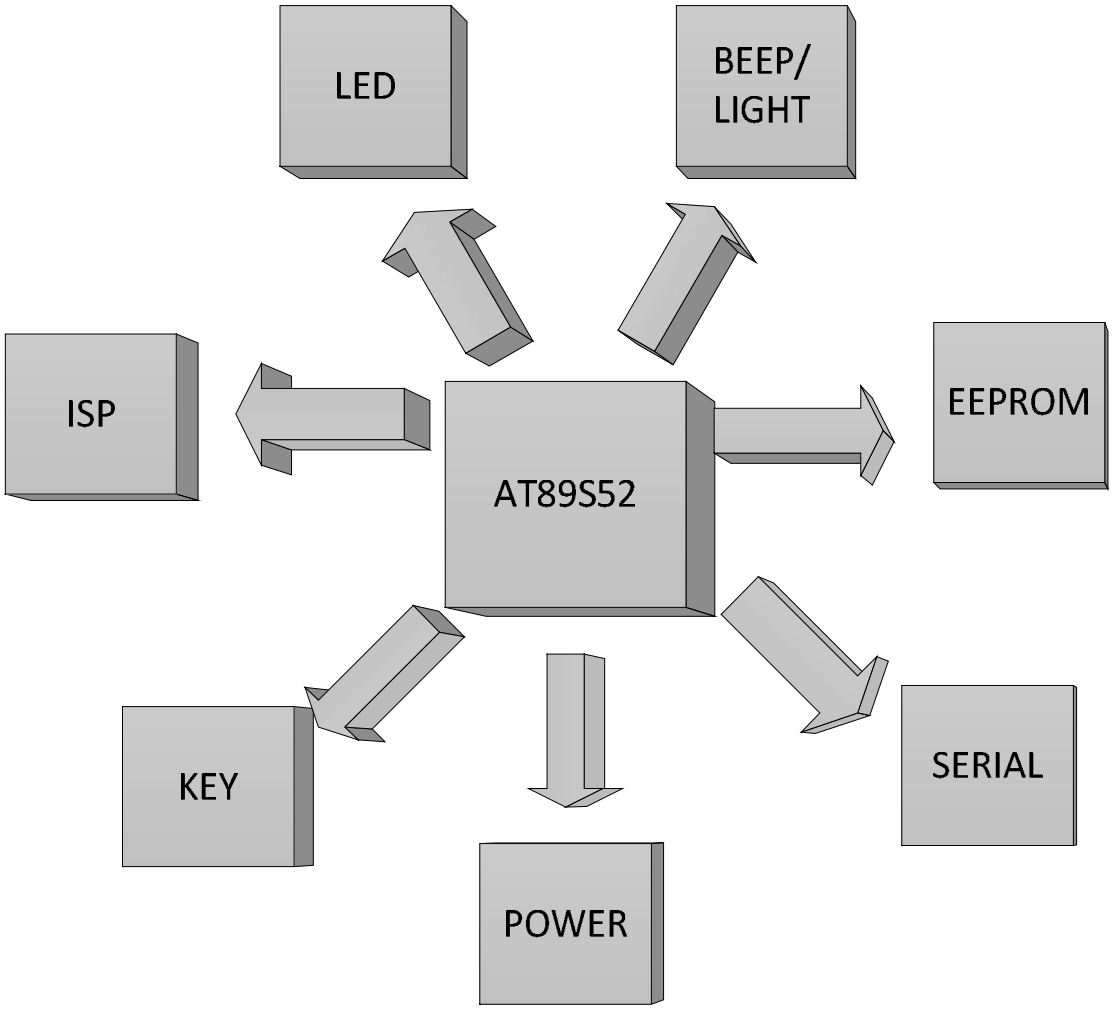


图 2-1 电子台签系统框图

输入部分：具有 7 个按键的键盘、用于模式切换的独立按键；

输出部分：16 个 8*8 的点阵屏、1 个电源指示灯、3 个模式指示灯和 1 个蜂鸣器

下载部分：串口下载；

存储部分：EEPROM；

电源部分：带有防反插功能的直流供电电路；

2.3单片机 I/O 引脚分配

经过详细的分析和设计，综合考虑功能需求、芯片及附加电路的具体情况、布局的便利性和合理性、以及管脚的复用情况，我们确定了 52 芯片的 I/O 管脚的分配。

以下是 I/O 引脚的分配情况：

表 2-1 I/O 引脚分配

模 块	单片机管脚
LED 点阵屏	P2.0 显示行选使能
	P2.1 行选 D
	P2.2 行选 C
	P2.3 行选 B
	P2.4 行选 A
7 键 键盘	P0.0-P0.6
EEPROM	P1.6 SCL
	P1.7 SDA
蜂鸣器	P1.0
LED 指示灯	P1.1 汉字显示
	P1.2 日期显示
	P1.3 倒计时显示
ISP 下载	P1.5 MOSI
	P1.6 MISO
	P1.7 SCK
串口通信	P3.0 RXD，复用串口数据
	P3.1 TXD，复用串口时钟
模式切换	P3.2 外中断

3 功能模块设计

3.1MCU 及其附属结构

AT89S52 要工作起来的最小系统需要有芯片，晶振，P0 口的上拉电阻，以及复位电路。

3.1.1 MCU 部分

主芯片 MCU：

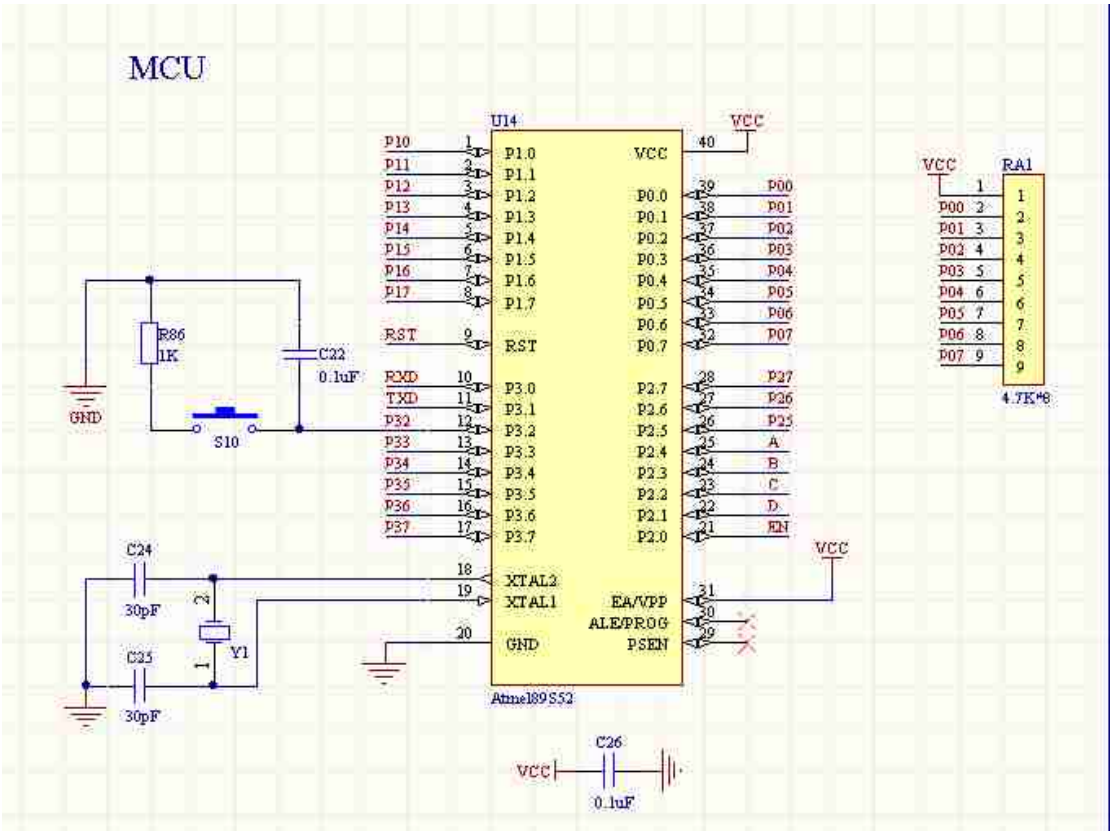


图 3-1 MCU 部分电路原理图

在 AT89S52 的四组 I/O 口中，P0 口比较特殊，是开漏极的双向 I/O 口，如果不外接上拉电阻，这个芯片是无法正常工作的。为统一电平驱动，Port 0 的 8 个引脚外接一个 4.7K*8 的排阻上拉到高电平。此时 P0 端口输出为低电平时的灌入电流约为 1mA（比较安全）。

单片机引脚 EA/VPP，PSEN 是外部存储器调用相关接口，EA 是调用存储器

的使能端，当它接高电平时单片机可以调用内部存储器，相反接低电平则调用外部存储器。PSEN 是外部存储器数据的输入端口。只有在需要外扩内存时才要用到。但外扩内存要损失大量的 I/O 口，故本实验板不考虑外扩存储，所以 EA 接高，PSEN 不做处理。引脚 ALE 是作为时钟外部输出，由于实验板不使用，所以也不做处理。

上图中左下角 18、19 号引脚 XTAL1、XTAL2 作为芯片内部时钟频率的驱动输入。本设计选用频率为 12MHz 的晶振作为频率输入源。晶振两边的滤波电容一般选择范围为 20～40pF，此处我们选择 30pF 可以完全满足需要得到稳定的输入。

上图中左上角 12 号引脚 P32 是单片机外中断 0 的输入，当开关 S0 被按下时，产生一个低电平，从而触发外部中断 0，在本系统中外部中断用于显示模式切换。

3.1.2 下载切换和复位开关

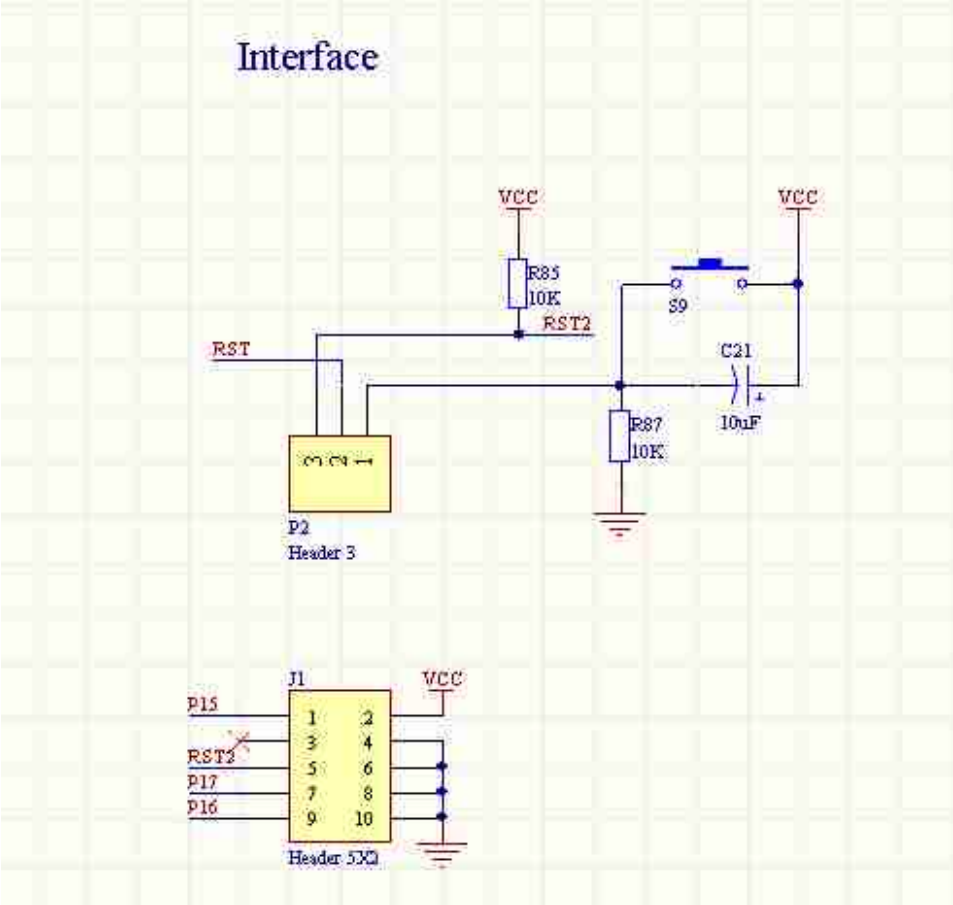


图 3-2 下载切换和复位开关电路原理图

最小系统的另一个部分是复位电路。之所以将复位电路和下载切换放在一起，是因为在下载的过程中要求复位管脚 RST 保持高电平，这时复位按键不能再用了。所以用一个双向带锁跳线开关 P2 控制下载和运行的切换。当开关 P2 接 23 时，连通外部下载模块的插针，可以进行串并口下载。开关 P2 接 12 时开发板正常工作。按键 S9 是复位开关，供电路复位使用。在处于下载状态时，将 RST 通过 R85 接到高电平，接 10K 的上拉电阻 R85 是防止灌入 RST 管脚的电流过大。

只要 RST 的引脚维持 2 个机器周期的高电平，MCU 就复位。R87 和按键 S9 组成上电延时电路和复位电路。上电时，晶振需要一定的起振时间，通过电容的充放电，使得单片机在一段延时后才开始工作，跳过晶振的起振时间。复位时，复位按键按下，电容开始充电和放电，能够维持一段时间的高电平，使单片机复

位。经过计算，此处选用 R87=10K，C21＝10uF。

3.2电源模块

电源模块采用外界 5V 供电，主要注重电压的稳定性，同时考虑了防反插等功能。原理图如下：

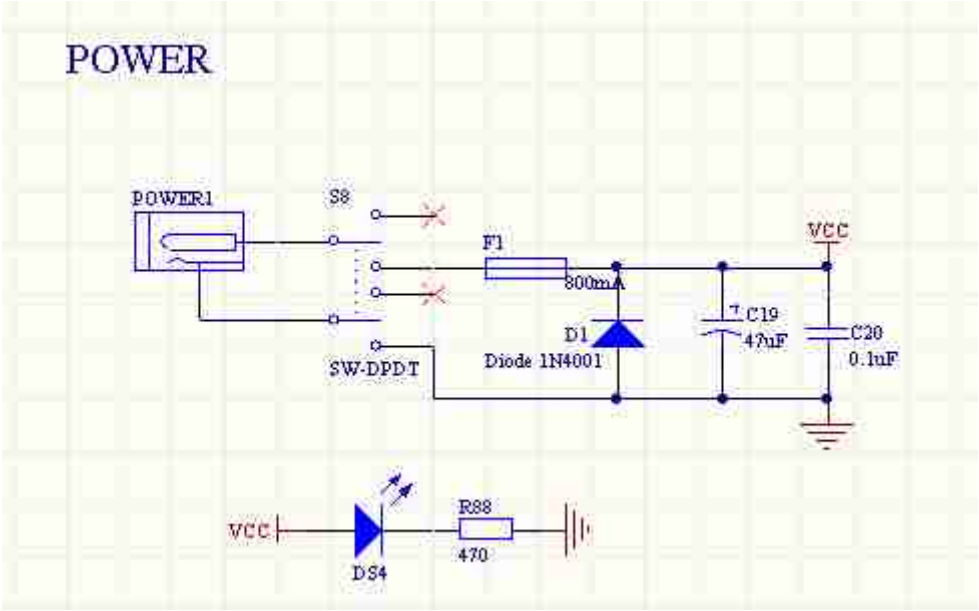


图 3-3 电源模块电路原理图

本电源模块使用发光二极管 DS4 作为电源指示灯。当电源接通时 DS4 发亮。由于发光二级管的承受电流一般在 3mA－10mA 之间，此处选择 470Ω 的电阻限流。

电源从图中 POWER1 处接入，通过双向带锁功能的开关（S8）作为电源的控制开关。这样可以在电源保持插入的情况下也可以很方便的控制好开发板的电源使用。F1 和二极管 D1 合起来组成一个防反接设备。F1 是一个带自恢复功能的保险丝，当电源正接时一切正常 D1 断路，反接时会因为瞬时电流过大而造成保险丝断开。当一定时间冷却后，保险丝又会自动连上从而电路恢复正常。保险丝的的最大承受电流从电路的整体功耗考虑选用 0.8A。

电容 C19 和 C20 是作为滤波使用。因为本电路板所需供电电压（5V）和功率都比较小，所以采用电容滤波可以完全满足要求。通过纹波过滤的基本关系 RC＝3～5T/2（其中 T 是纹波周期，R 为开发板输入电阻，约为 50 欧）。本开发

板选用 0.1uF 和 47uF 电容分别对 10KHz 以下和 10M～20MHz 的毛刺波纹有比较好的滤波效果。实际测得采用以上电路输入电压的波动平均小于 10Hz。

3.3LED 点阵屏

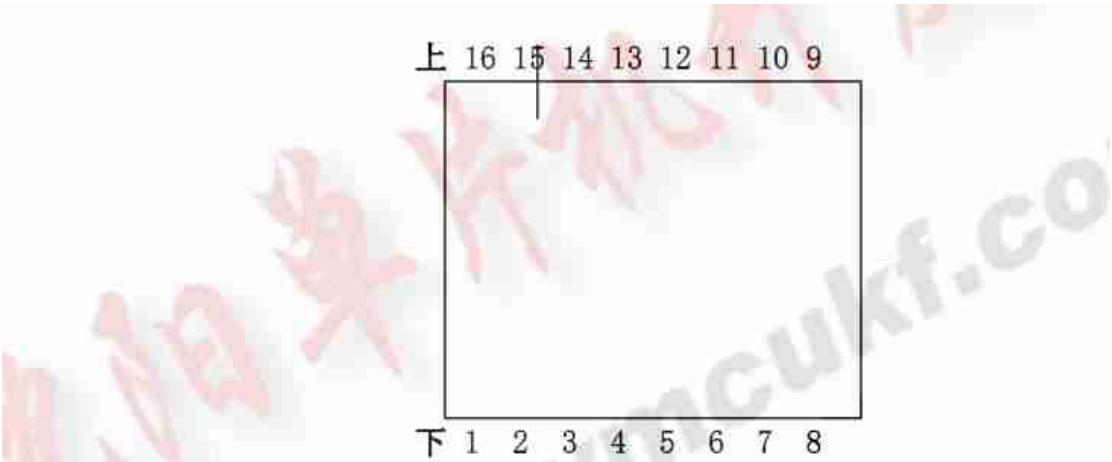
3.3.1 LED 点阵屏介绍

8*8 的 LED 点阵为单色行共阴模块,单点的工作电压为正向(Vf)=1.8 v ,正向电流(IF)= 8-10mA。静态点亮器件时(64 点全亮)总电流为 640mA,总电压为 1.8 v, 总功率为 1.15 W。动态时取决于扫描频率(1/8 或 1/16 秒),单点 瞬间电流可达 80-160 mA。

16*16 点阵静态时 16*16*10mA,动态时单点电流 80-160mA。

接线方式:

- 当某一行线打高时：
  - 某一列线为低时,其行列交叉的点就被点亮;
  - 某一列线为高时,其行列交叉的点为暗。
- 当某一行线打低时,无论列线如何,对应这一行的点全部暗。



1	控制	第五行显示	接高	9	控制	第一行显示	接高
2	控制	第七行显示	接高	10	控制	第四列显示	接低
3	控制	第二列显示	接低	11	控制	第六列显示	接低
4	控制	第三列显示	接低	12	控制	第四行显示	接高
5	控制	第八行显示	接高	13	控制	第一列显示	接低
6	控制	第五列显示	接低	14	控制	第二行显示	接高
7	控制	第六行显示	接高	15	控制	第七列显示	接低
8	控制	第三行显示	接高	16	控制	第八列显示	接低

图 3-4LED 点阵屏的接线方式

3.3.2 电路连接

3.3.2.1控制系统的逻辑结构

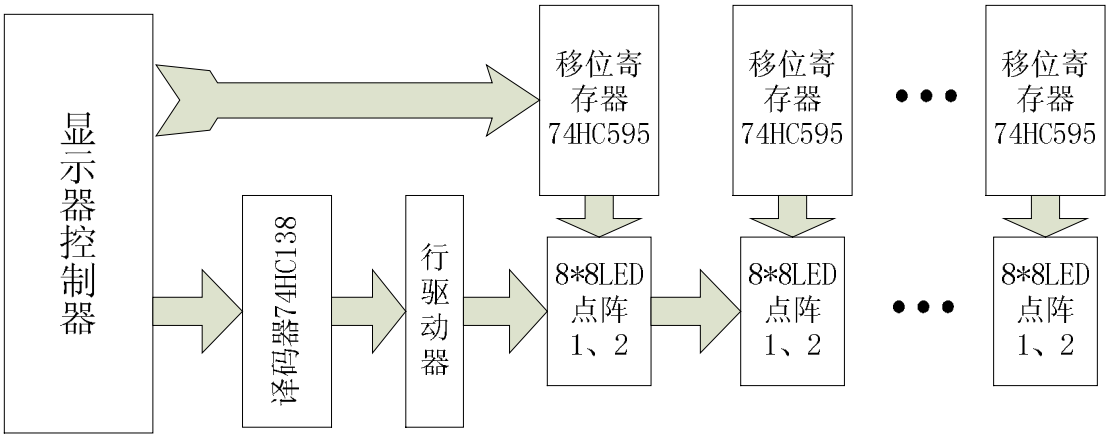


图 3-5LED 点阵控制系统逻辑结构

如上图所示，点阵显示屏每个单元由 16 个 8×8 点阵 LED 显示模块、行信号选择译码器 74HC138、数据移位寄存器 74HC595、行驱动器组成。16 片 8

×8 点阵 LED 显示模块组成一个 64×16 的 LED 点阵屏，用于同时显示 4 个 16×16 点阵汉字或 8 个 16×8 点阵的汉字、字符或数字。单元显示屏可以接收来自控制器(主控制电路板)或上一级显示单元模块传输下来的数据信息和命令信息，并可将这些数据信息和命令信息不经任何变化地再传送到下一级显示模块单元中, 因此显示板可扩展至更多的显示单元,用于显示更多的显示内容。

3.3.2.2LED 点阵的引脚接线

本系统使用的 LED 的引脚接线如下所示，显示了其引脚对应关系：

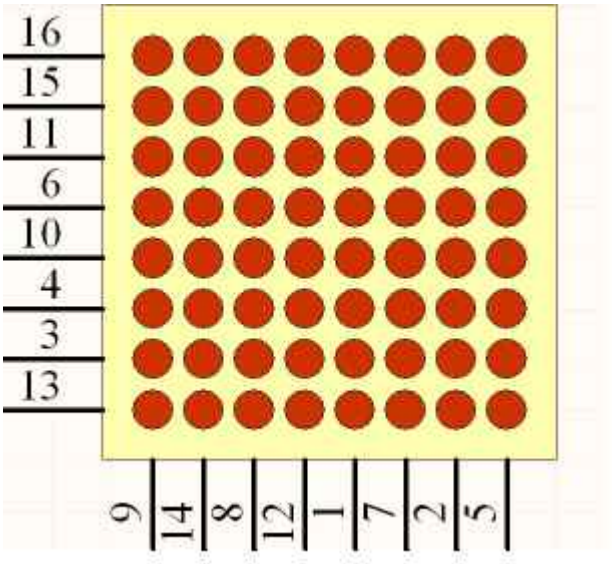


图 3-6LED 点阵的引脚接线

3.3.2.3单汉字显示单元

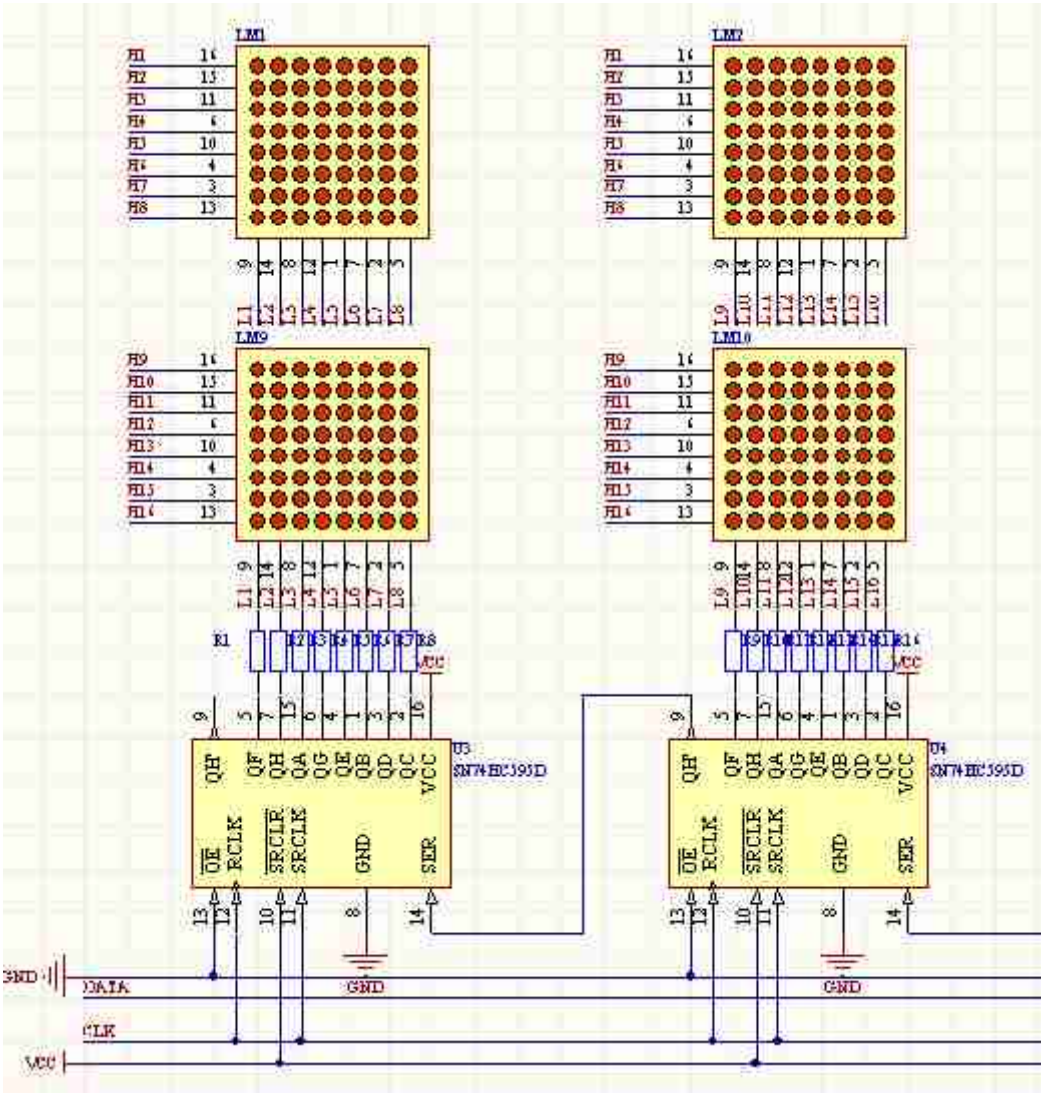


图 3-7 单汉字显示单元

如上图所示，用 4 个 8*8 的点阵屏显示一个汉字，通过 74HC595 的移位寄存来控制每一列的 LED 的亮灭。行选是通过 74HC138 来控制的，将在下一节中说明。通过行选和列选的同时作用，来控制某一个点的点亮与熄灭。

3.3.2.4LED 点阵行选

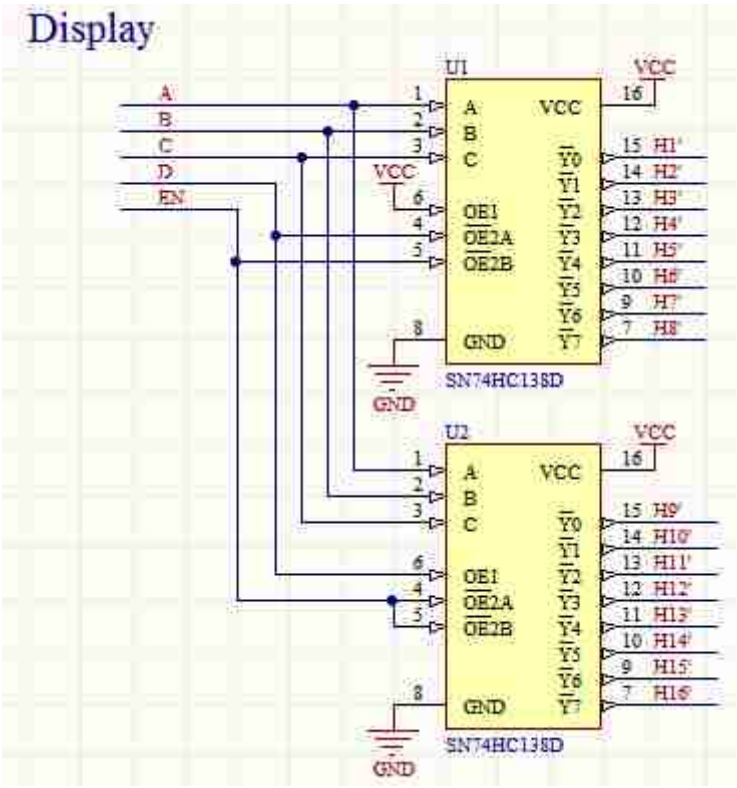


图 3-8LED 点阵行选

如上图所示，通过两个 74HC138 芯片扩展为 4-16 线译码器，通过使能端 EN 控制显示屏的显示与否，通过 A/B/C/D 四个行选信号决定点亮哪一行。

3.3.2.5LED 点阵行驱动器

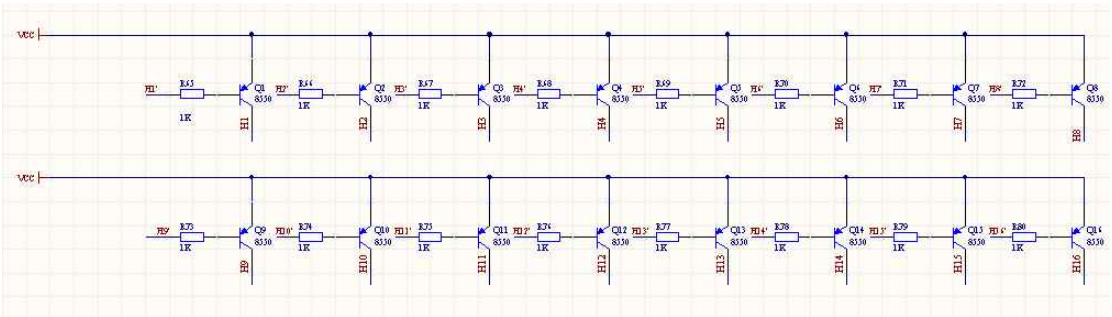


图 3-9LED 点阵行驱动器

如上图所示，本系统应用 PNP 三极管 8550 来作为 LED 点阵的行驱动器，8550 工作在开关状态，LED 发光二级管的压降约为 1.5V，经计算基极电阻取 1k。

3.3.2.6LED 点阵列选

如图 3-7 单汉字显示单元所示，通过 74HC595 的移位寄存实现列选功能。其中移位寄存器的数据和时钟通过复用串口模式 0 来实现，这点讲在串口通信这一节介绍。

3.4蜂鸣器与 LED 模块

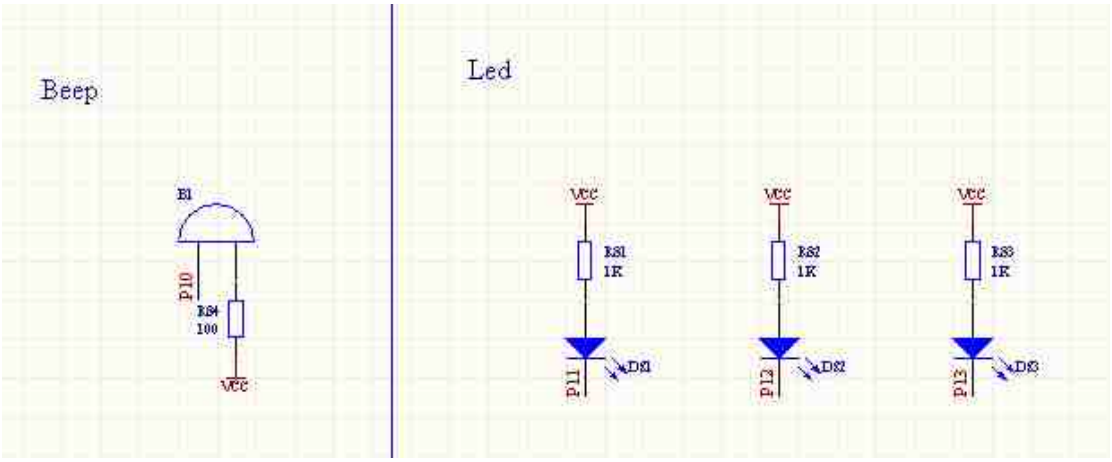


图 3-10 蜂鸣器与 LED 部分电路连接图

有源蜂鸣器：工作电压 0～5V，电流 5mA。电阻防止蜂鸣器短路时电流过大损坏 I/O 口，取 100 欧姆。

蜂鸣器的负极接在 P1.0 口上，正极经过一个电阻接在电源上，所以只有 P1.0 输出低电平时才会发出声音。上电时，P1.0 输出高电平，不会发出声音。当 P1.0 口输出一定频率的脉冲矩形波时，根据频率的不同，可以产生不同的音调，进而可以产生乐曲。人的听觉范围是（15～20KHz）。

LED：工作电压 0～5V，压降 0.7V，工作电流 3～5mA，电阻取 1K。

LED 通过外中断控制，每按一次外中断按键，P32 口发出一个低电平，此时通过软件控制一个 LED 点亮，从而指示当前的工作模式。

3.5键盘模块

本系统使用了 7 键键盘，由于有足够空闲 I/O 口，故采用简单按键的接线方

式，通过 7 个 IO 口来对 7 个按键进行扫描。具体电路连接如下图：

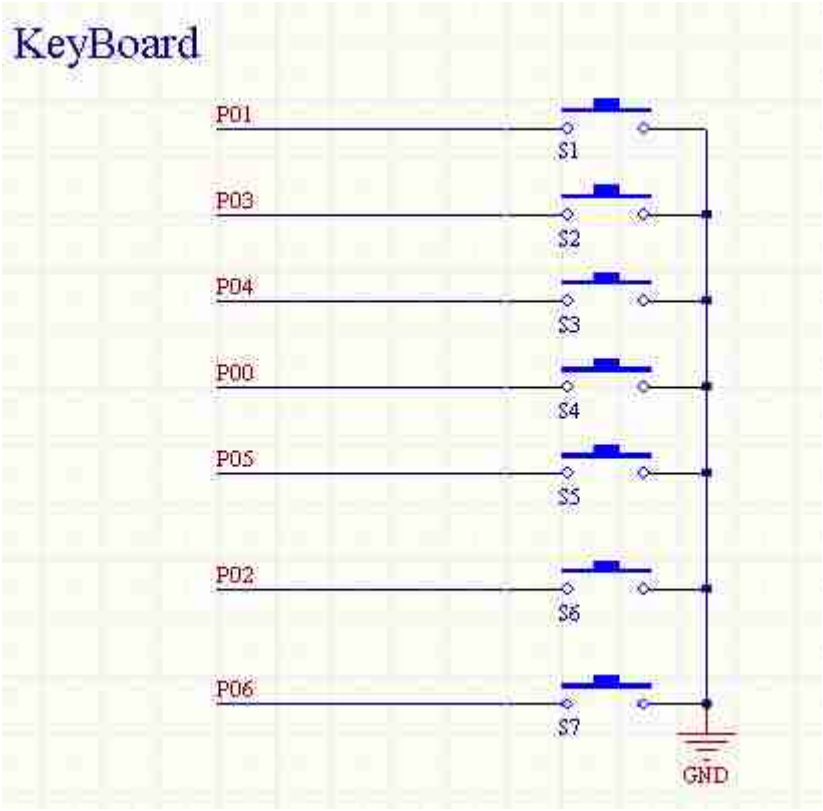


图 3-11 键盘电路连接

采用简单按键，一方面简化了硬件连线，另方面也使软件的键盘扫描部分变的容易了许多。通过编程可以实现单键的短按和长按，单击和双击，以及组合键。

3.6 EEPROM

3.6.1 芯片介绍

本系统采用的器件为 AT24C04，存储空间为 4KB，可擦写次数为 1 百万次。管脚分布如下：

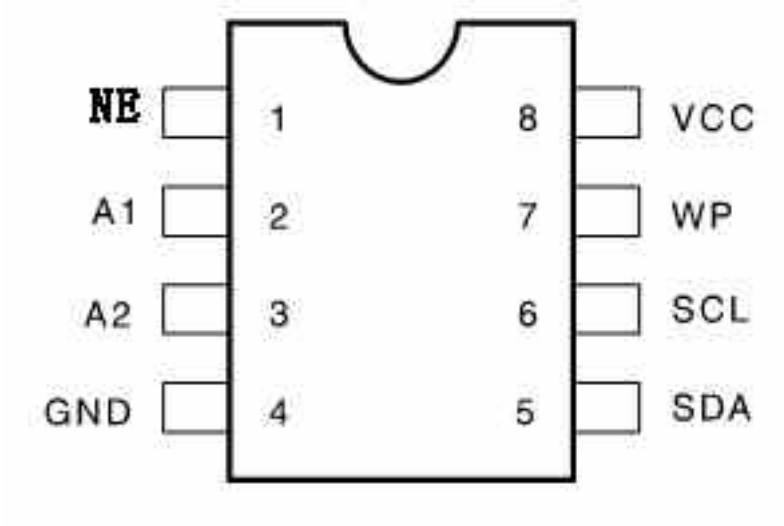


图 3-12 AT24C04 封装图

管脚 NE 表示无连接，A1、A2 为片选，作为 IIC 总线上该器件的标示。管脚 7 为写保护。SCL、SDA 分别为时钟线和控制线。

EEPROM 为 IIC 器件，读写时序服从 IIC 协议 (请参照附录 I2C 总线的原理)。

3.6.2 电路连接

部分电路连接如下图所示：

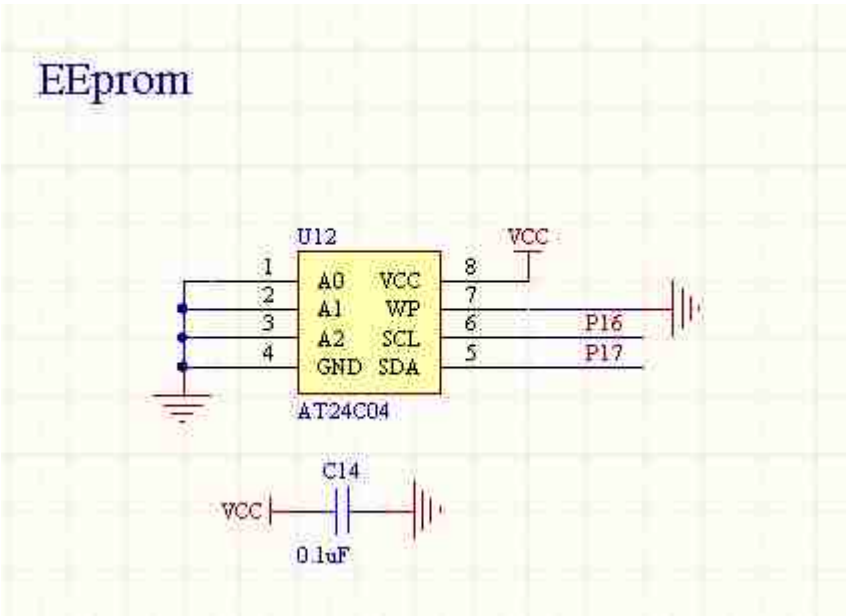


图 3-13 EEPROM 部分电路连接

将引脚 1 接地而不是悬空有利于实验板的兼容性，当 EEPROM 更换为 AT24C02 或其它器件时同样可以控制。A1、A2 接地表明了 EEPROM 的地址为

00。WP 接地表示芯片可写。

时钟线(SCL)、数据线(SDA)分别连接在 P1.6 和 P1.7 上，电容 C14 为 AT24C04 的滤波电容。

3.7 串口通信

3.7.1 芯片介绍

本模块采用 MAX232 芯片，将单片机 RXD\TXD 两个端口输出的 TTL 电平与 RS232 通信线路的电平进行转换。芯片的引脚图如下：

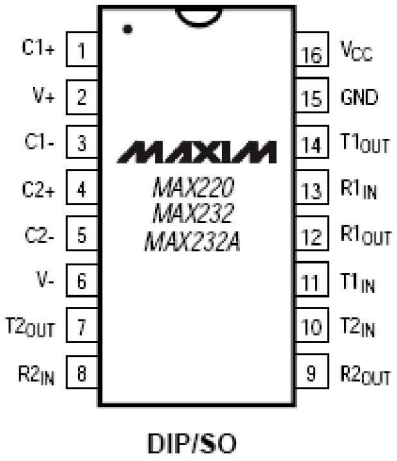


图 3-14 MAX232 引脚功能图

MAX232 总共含有两组（4 个）电平转换通道，可以独立实现四条线路的电平转换。参考 Datasheet 上的说明，MAX232 的连接原理图如下：

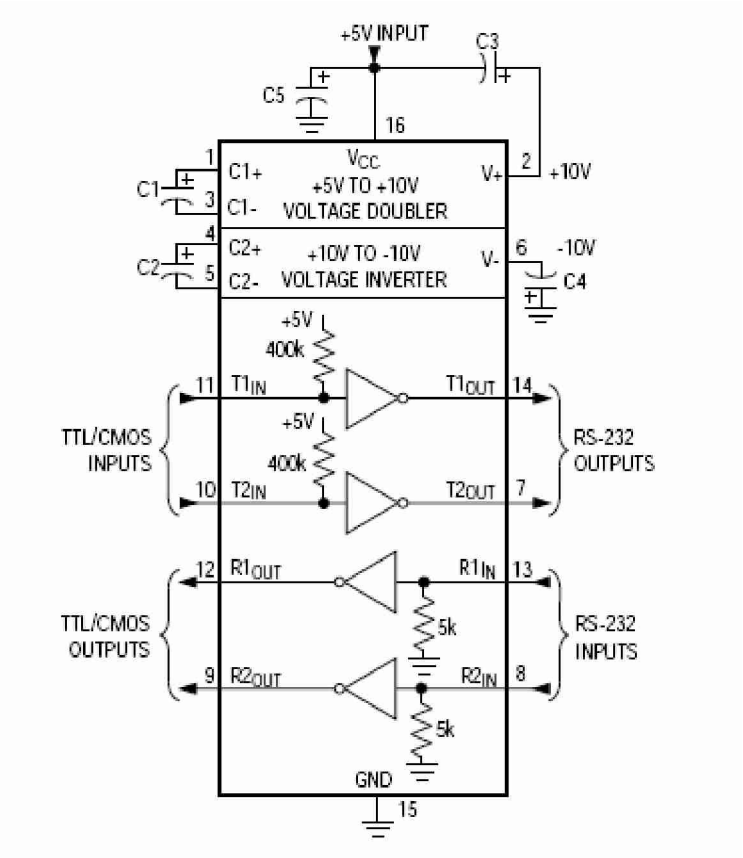


图 3-15 MAX232 引脚连接原理图

MAX232 的外围电路总共包含 5 个电解电容，需要注意的是，2(V+)、6(V-)的电容连接方向与其他电容的连接方向不同，其电压参考方向是实际电压的相对大小（+10V 与 VCC、-10V 与 GND）。

3.7.2 串口复用

为了严格控制 LED 的列选信号，并且保证列选信号的及时显示，我们需要快速的寄存数据，并获得 74HC595 的精确时钟，于是设计了通过复用串口模式 0，通过模式 0 的 TXD 口来发送数据，RXD 口则输出严格的时钟控制（1/12 晶振频率）。

而在串口模式 1 下，则通过上位机下载点阵数据到单片机中。

串口的模式选择通过软件控制使能端实现，默认开机选择模式 0，即显示点阵。

3.7.3 电路连接

MCS-52 内部有一个可编程的串行端口，可以实现全双工（Full Duplex）通信。通过串行端口，可以使用较少的传输线完成数据的传输，实现与其它单片机或 PC 的通信。

本系统中串口模块由 MAX232、9 针串口（请参照附录串口针脚定义）、 外围电容组成，原理图如下：

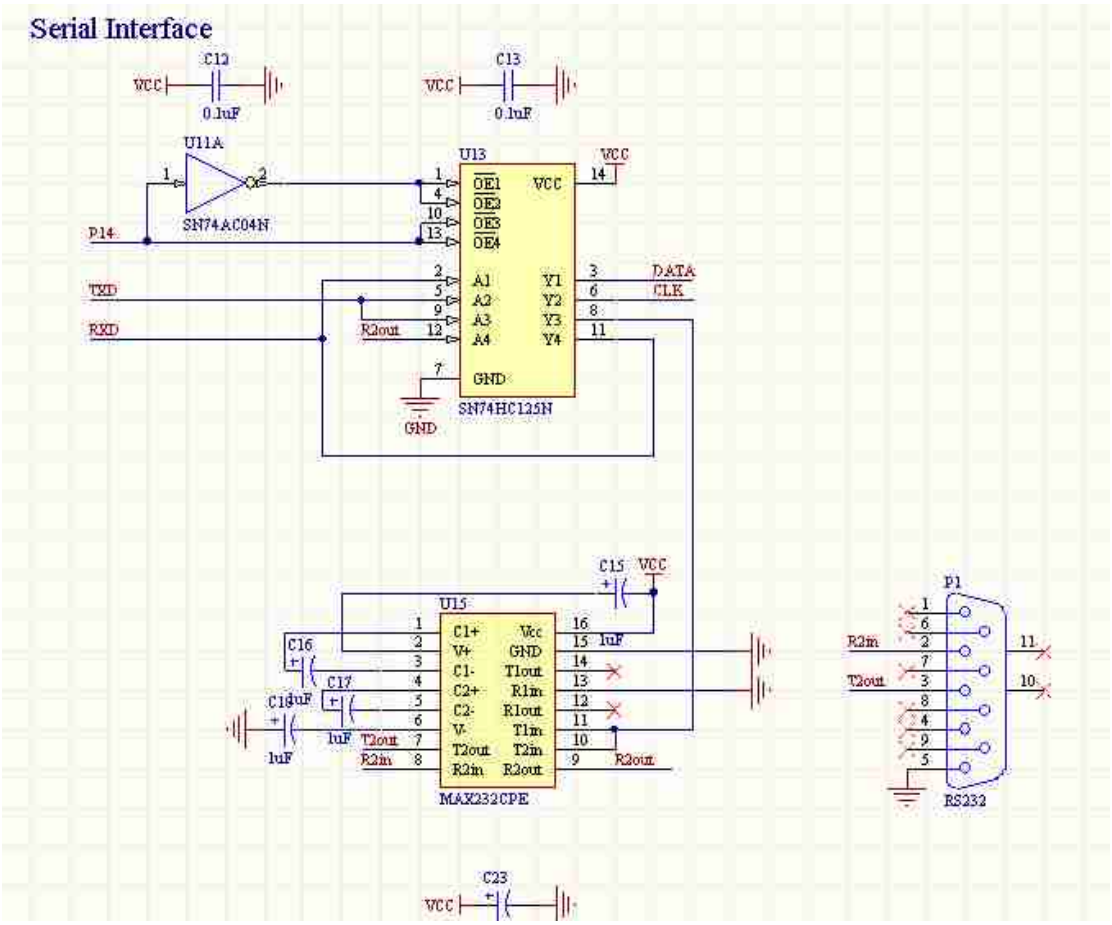


图 3-16 串口模块原理图

其中，串口的 2、3 端口分别与 MAX232 的 R2IN 和 T2OUT 相连，串口 5 脚接地。另外，VCC 和 GND 之间经接了电解电容 C23，作为滤波电容。

在传输数据模式下，MCU 的 RXD 与 TXD 通过 74HC125 三态门分别接至 74HC595 的数据端和时钟端。

在下载模式下，MCU 的 RXD 与 TXD 通过 74HC125 三态门分别与 MAX232

的 R1OUT 和 T1IN 相连。

两种模式的选择通过控制 I/O 口 P14 实现，由软件控制。

3.8 ISP 下载

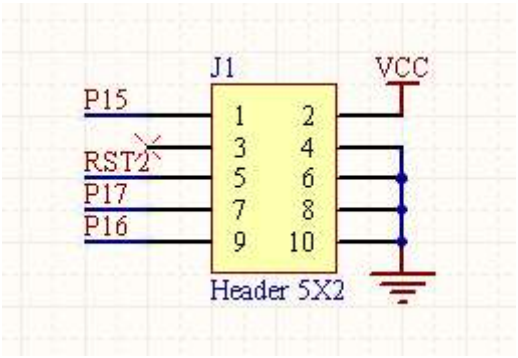


图 3-17ISP 下载电路

ISP（In-System Programming）是 AT89S52 单片机提供的程序下载方式，主要通过 MCU 的 P1.5(MOSI)、P1.6(MISO)、P1.7(SCK)和 RST 引脚的电平控制来实现。其中，MOSI 为 PC 的数据输入端，由 MCU 接收来自 PC 的编程数据；MISO 为 PC 的数据输出端，有 MCU 向 PC 发送反馈数据，用于数据确认和校验；SCK 为时钟线，通过 PC 端送入周期变化的电平来控制编程时序；RST 为重置端，用于在编程时对 MCU 的状态进行重置。

由于在进行每一个实验前都需要烧写程序，所以下载部分是不可或缺的一个重要部分，将下载部分放在主板上会更方便使用。在系统上仅提供串口下载一种方式。

4 上位机软件设计

4.1 概述

为使用户更为方便的生成汉字和绘制自己的图形，我们为《电子台签》开发了一款跨平台的上位机软件。

本软件可以自动生成常用汉字字模，也可以方便的绘制直线，最终能将这些

图像信息通过串口发送至下位机。

本软件使用 QT 框架编程实现，并考虑了 Windows 和 Unix 系统的区别，是一个跨平台的软件项目。此外，我们使用 QSerialDevice 这个第三方库来封装对串口的操作。

在软件的工程搭建上，我们采用了工程文件和源文件分离的管理方式，将工程文件从源文件目录中分离出来，单独构成目录。并且，提供了 vs2008 工程以及 bat 文件脚本以方便 Windows 下的开发和调试。

在软件构架上，我们尽力遵循“低耦合，高内聚”的编程原则，将各个功能模块独立成类，方便维护和扩展。

4.2 总体设计

上位机最重要的功能就是方便的构造显示数据，因此，其设计都是围绕数据的处理和显示来进行的，总体的框架如下：

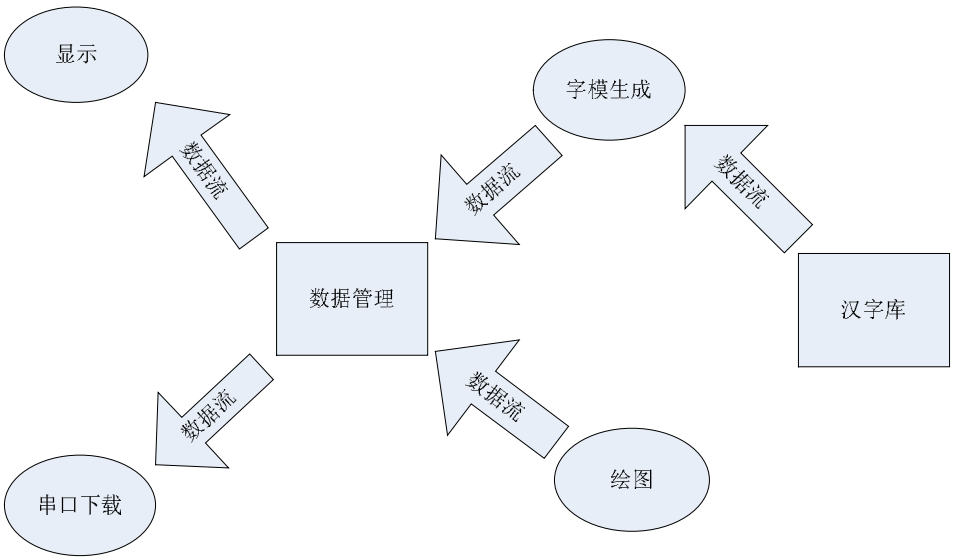


图 4-1 上位机软件总体框架

从图中我们看到，数据的来源有两个，一个是字模的生成，一个是绘图。

因为我们的产品是“电子台签”，所以我们的用户最常用的功能必然是汉字显示。因此，我们提供了一个汉字库，程序可以方便的从该库里面获取字模信息，并更改点阵数据。

另一方面，绘图（绘制直线）也能更改点阵数据。使点阵的生成更灵活。所有数据的更改都会立即被显示出来，方便查看。而数据最终可以通过串口下载发送到单片机中，在电子台签上显示。

4.3 模块设计

4.3.1 模块概述

根据总体设计的思想，构建如下的功能模块：

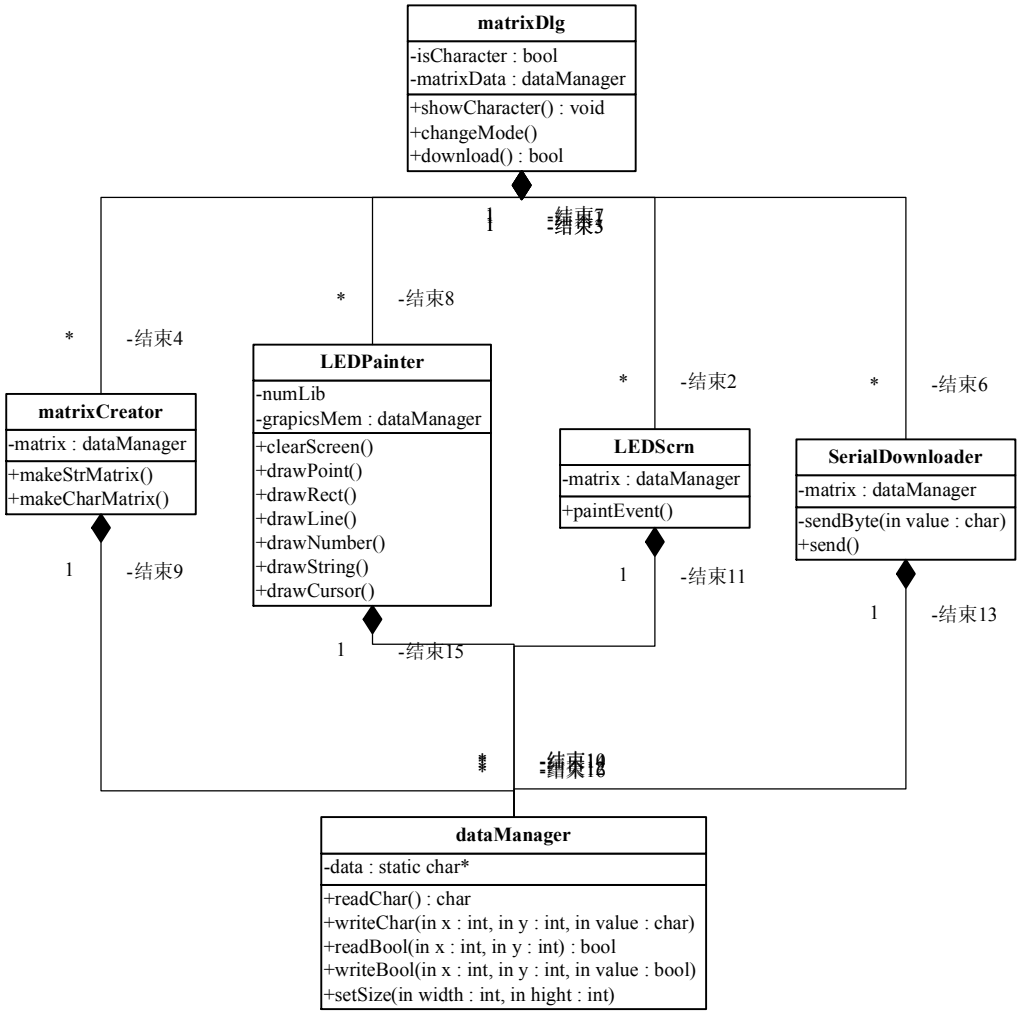


图 4-2 上位机软件模块划分

其中，DataManager 即负责数据的管理，包括数据按位读写，按字节读写等功能。

MatrixDlg 负责整个系统的管理和操作，也是软件的用户接口。

MatrixCreator 和 LEDPainter 进行数据的写操作，构造点阵数据。

LEDScrn 则负责将数据显示到 MatrixDlg 上。

SerialDownloader 是软件的下载模块，负责将数据串行发送到计算机串口。

4.3.2 数据管理模块

DataManager 是数据的管理模块，负责为数据的读写提供接口。

DataManager 中的数据按字节存储，又包含按位读写操作，字节与位之间的转换算法较为负责，另一方面，这种对数据的管理操作很常见，因此我们将着重介绍这个类的设计方法。

该类含有以下接口：

```
/*
 *     DataManager(int width= 8, int height= 16)
 *         constructor
 *         create a data manager with width and height (default is 8 x 16).
 *     unsigned long getLength()
 *         get length of this data container.
 *     void writeChar(int x, int y, unsigned char cValue)
 *         write a unsigned char data to the position (x, y) in matrix.
 *         note that the data location is description by char count.
 *     void writeBool(int x, int y, bool bValue)
 *         write a bool data to the (x, y) in matrix.
 *         note that the data location is description by bool count.
 *     unsigned char readChar(int x, int y) const
 *         read a char data from (x, y)
 *     bool readBool(int x, int y) const
 *         read a char bool from (x, y)
 *     void clearAll()
 *         Set all data to 0.
 */
```

这些接口中，包含了对数据的按位、按字节读写接口，可以方便的绘图（类位图形式），也可以方便的进行数据传输（字节传输）。

其底层是使用 *char* 型数组存储的，因此按位的操作算法较为复杂，包含各种

对字节中位的操作，下面以 *readBool* 为例说明位与字节的转换算法。

函数原型如下：

```
// Read and write value using bool type
bool DataManager::readBool(int x, int y) const
{
    int chCount= x/8;            // switch to the x-index of char
    int restBit= x%8;

    // current char = matrixData[chChount][y]
    char curChar= *(matrixData + (dataWidth*y + chCount));

    // current bit is at (1000 0000 >> restBit)
    // current bit larger than 0, return true.
    return curChar & (0x80 >> restBit) ? true : false;
}
```

该函数首先计算出 *x,y* 坐标对应的字节位置（*int chCount= x/8;*）和余位（*int restBit= x%8;*），接着读取对应字节，然后取出该字节中正确的位。

最后一部，取出正确的位，是该函数的重点。

若字节中的数据如下：*abcdefgh*，则当 *restBit* 为 3 时，需取出 d。那么先使 1000 0000 右移 3 位 *(0x80 >> restBit)* 成为：0001 0000，接着和当前的字节做与操作 *curChar & (0x80 >> restBit)*，成为：000d 0000，返回值则取决于 d 的值了。

该算法的优势在于使用移位和与操作来获取正确的位，效率很高，可以移植到单片机程序中。

其他函数比较简单，不再赘述。

4.3.3 字模生成模块

MatrixCreator 是字模生成模块，调用 HZK16 文件的字模构造点阵内容。

该类依赖于上文说到的 DataManager 类。

接口如下：

```
/*
```

```
 *  bool makeStrMatrix16(char const *wstr, unsigned int const wideCharLength,
DataManager *data);
 *      input a string and get the string's matrix data to data.
 *  bool makeCharMatrix16(unsigned char const word[], char *wordData);
 *      input a char array (2 char a chinese word) word and get the char's
matrix data to wordData.
 */
```

一般来说，该类对外的接口只需要 *makeStrMatrix16* 就够了，该函数的作用是将一个字符串转换为字模的点阵数据，算法比较简单，不做累述，下面介绍 *makeCharMatrix16* 的算法。

函数原型如下：

```
bool MatrixCreator::makeCharMatrix16(unsigned char const word[], char
*wordData)
{
    // get the word's zone/bit code
    unsigned char zoneCode= word[0] - 0xa0;
    unsigned char bitCode = word[1] - 0xa0;
    long offset = (94*(zoneCode-1) + (bitCode-1))*32;

    std::ifstream infile(matrixLibPath.c_str());
    if (!infile.is_open())
    {
        return false;
    }

    // get 32 byte data from position 'offset'.
    infile.seekg(offset, std::ios_base::beg);
    if (!infile.good())
    {
        infile.close();
        return false;
    }
    infile.read(wordData, 32);
    infile.close();
    return true;
}
```

最重要的一点，是需要弄清楚 HZK16 文件的字模储存格式。

HZK16 是依靠汉字的区位码来定位汉字字模的，函数首先取出汉字的区位

码：

```
// get the word's zone/bit code
unsigned char zoneCode= word[0] - 0xa0;
unsigned char bitCode = word[1] - 0xa0;
```

然后计算出该汉字在 HZK16 文件中的偏移量：

```
long offset = (94*(zoneCode-1) + (bitCode-1))*32;
```

接着，打开文件，读出该偏移量后 32 字节的内容，即为字模信息：

```
// get 32 byte data from position 'offset'.
infile.seekg(offset, std::ios_base::beg);
if (!infile.good())
{
    infile.close();
    return false;
}
infile.read(wordData, 32);
```

4.3.4 绘图模块

LedPainter 是软件的绘图模块，通过对点阵数据的操作绘制各种图形。

接口如下：

```
// must initial with graphics memory pointer.
LedPainter(DataManager *);

// clear the screen, make all led off.
void ClearScreen();
// draw point using method.
void DrawPoint(GraphVector const, enum PaintMethod);
void DrawPoint(int const, int const, enum PaintMethod);
// using the paint method to draw a rect/line at top-left, button-right
area.
void DrawRect(GraphVector const, GraphVector const, enum PaintMethod);
void DrawLine(GraphVector const, GraphVector const, enum PaintMethod);
// draw a number place it's top-left conner on specific location.
void DrawNumber(GraphVector const, byte const, enum PaintMethod method
= PM_COPY);
// draw a number(time) string on specific location (top-left).
void DrawString(GraphVector const, byte const[], byte, enum PaintMethod
method = PM_COPY);
// draw a cursor on specific location using NOT method.
```

```
void DrawCursor(GraphVector, byte size = 3, enum PaintMethod method =
PM_XOR);
```

该函数具备各种图形的绘制功能，并且一般都含有异或和直接绘制两种模式。

该模块的函数较为简单，不做累述，想了解其实现，请直接查看其源码。

4.3.5 显示模块

LEDScrn 是显示模块，该模块的作用很简洁明了，就是将点阵数据呈现到对话框上。

其接口只有 2 个：

```
QLEDScrn(DataManager *data, QWidget *parent= 0);
void paintEvent(QPaintEvent *event);
```

初始化时指定显示的数据，之后在需要绘制图像时，QT 框架会自动调用重载函数 *paintEvent*。

下面以该模块为例，简要介绍一下 QT 框架的图形绘制。

QT 中，简单的图形绘制可以使用 QPainter 实现，初始化时指定其 parent。接着可以使用 setBrush 和 setPen 设置刷子（背景）和钢笔（前景）。

QPainter 的绘制可以是基于直角坐标系的。默认的坐标原点在其父窗口的左上角。X 轴正方向向右，y 轴正方向向下，单位为像素。

值得注意的是，QPainter 可以通过 translate 来移动坐标原点，如 translate(3,4) 可以将坐标原点向右移动 3 个单位，向下移动 4 个单位。

除了这些基本的设置外，还有很多其他的设置，详情请参考 QT 的官方文档。

QPainter 除了设置，当然还有绘制的功能，QPainter 提供大量的绘图函数方便图形的绘制，具体情况不做详述，下面请看本软件中的显示函数来体会：

```
void QLEDScrn::paintEvent(QPaintEvent * /*event*/)
{
```

```
    QPainter painter(this);

    float xSpace= (float)width()/LED_MATRIX_X_COUNT;
    float ySpace= (float)height()/LED_MATRIX_Y_COUNT;
    float const scale= 6.0;
    QRect rect(xSpace/scale, ySpace/scale, xSpace-xSpace/scale,
ySpace-ySpace/scale);

    // draw black background
    painter.setBrush(Qt::SolidPattern);
    painter.drawRect(QRect(0, 0, width() - 1, height() - 1));

    for (int x= 0; x < LED_MATRIX_X_COUNT; x++)
    {
        for (int y= 0; y < LED_MATRIX_Y_COUNT; y++)
        {
            painter.save();
            painter.translate(x*xSpace, y*ySpace);

            if (matrixData->readBool(x, y))
            {
                painter.setPen(Qt::red);
                painter.setBrush(Qt::red);
            }
            else
            {
                painter.setPen(QPen(Qt::red, 0, Qt::DotLine));
            }

            painter.drawEllipse(rect);      // draw ellipse use specific style

            painter.restore();
        }
    }
}
```

该函数的作用是按位遍历点阵数据，为 1 则绘制一个红色小圆盘，否则绘制红色虚线圈。函数循环中，使用 `painter.save();`和 `painter.restore();`来保存和恢复原点、颜色等信息。为循环内部的坐标、颜色变化提供方便。

4.3.6 串口下载模块

SerialDownloader 是串口下载模块，该模块可以将软件中的点阵数据通过第三方的串口操作库发送到计算机的串口。

其接口如下：

```
/*
 *  bool openDevice(const QString &dn);
 *      Open the serial device according to device name 'dn'.
 *  bool sendBytes(QByteArray ba);
 *      Send QByteArray data to serial.
 *  void close(void);
 *      Close serial device.
 */
```

该模块实际上是对第三方库 qserialdevice 的封装。实现很简单，在初始化串口设备时，可以指定串口的波特率、数据长度、验证模式、停止位等配置：

```
if (serialPort->isOpen())
{
    if (!serialPort->setBaudRate(AbstractSerial::BaudRate9600)) {
        return false;
    };
    if (!serialPort->setDataBits(AbstractSerial::DataBits8)) {
        return false;
    }
    if (!serialPort->setParity(AbstractSerial::ParityNone)) {
        return false;
    }
    if (!serialPort->setStopBits(AbstractSerial::StopBits1)) {
        return false;
    }
    if (!serialPort->setFlowControl(AbstractSerial::FlowControlOff)) {
        return false;
    }
}   // if (serialPort->isOpen())
else
    return false;
```

通过串口名称（Windows 下为 COMn，Unix 下为/dev/ttySn）打开串口：

```
serialPort->setDeviceName(dn);
if (serialPort->open(AbstractSerial::WriteOnly))
```

```
    return true;
```

然后可以通过写串口发送数据：

```
if (ba.length() == serialPort->write(ba))
    return true;
```

4.3.7 对话框模块

MatrixDlg 是对话框模块，该模块是本软件的用户接口，也是整个软件的控制者，它组合了以上各个模块，完成用户指定的功能。

接口函数如下：

```
void comitContents();
void startDownload();
void changeSerial();
```

界面如下：

图 4-3 软件界面

点阵显示区是绘图模块的父窗口，绘图模块的更新将显示于该区。

【提交】对应 `comitContents` 函数，负责将汉字输入框中的汉字转换成点阵后设置到点阵数据中；

【下载】对应 `startDownload` 函数，负责将点阵数据由串口发送出去；

设备名称后的下拉框内容的更改将触发 `changeSerial` 函数，更改指定的串口

设备。

【画线】功能暂未实现。

该类中函数的算法也比较简单，可以直接通过源代码了解其实现，这里不再累述。

4.4 实际显示效果

4.4.1 Windows 7 下的效果

图 4-4 Windows 7 下的显示效果

4.4.2 Ubuntu 10.10 下的效果

图 4-5 Ubuntu 10.10 下的显示效果

5 附录

5.1 I^2C 总线的原理

I^2C 总线是二线制总线，I^2C 总线由一条串行数据线 SDA 和一条串行时钟线 SCL 组成。SDA 和 SCL 分别是 24LCxx 的数据输入输出端和时钟输入端。24LCxx 的位数据通过 SDA 引线输入输出。在 SCL 为低电平期间待传输的数据要出现在 SDA 引线上，SCL 为高电平期间 SDA 引线上的数据位要稳定有效。一个 SCL 脉冲传送一位数据。SDA 和 SCL 为漏极开路端，使用时需要接上拉电阻。

下面分别是 IIC 总线时序图和 IIC 总线数据检验时序图。

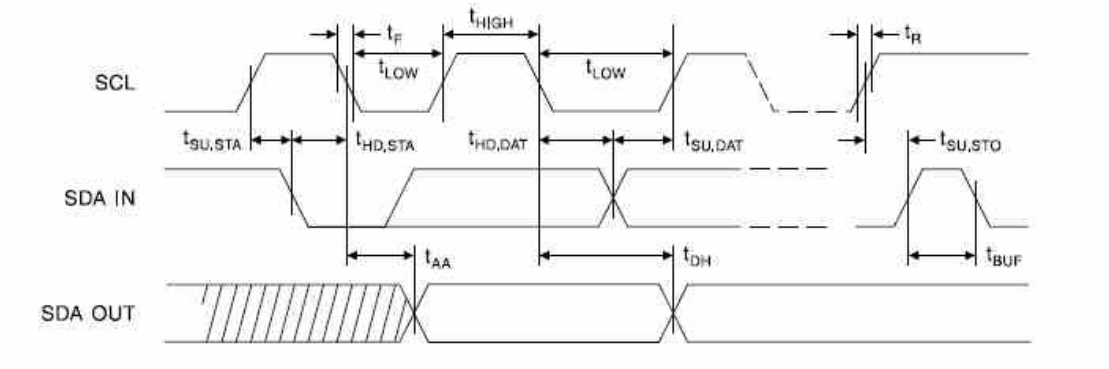


图 5-1 IIC 总线时序图

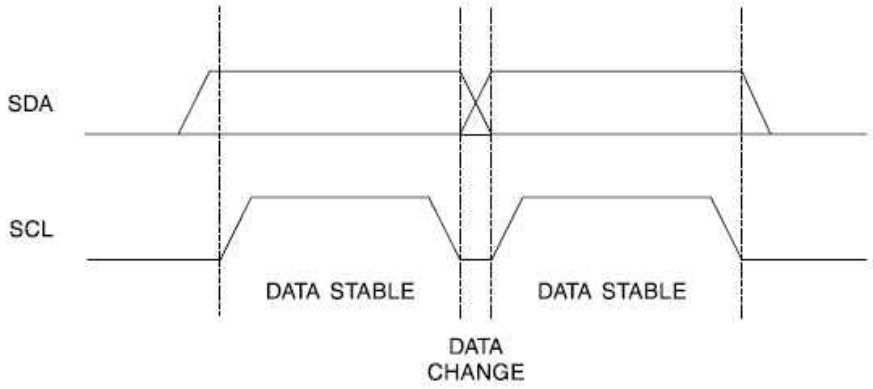


图 5-2 IIC 总线数据检验时序图

5.2 串口针脚定义

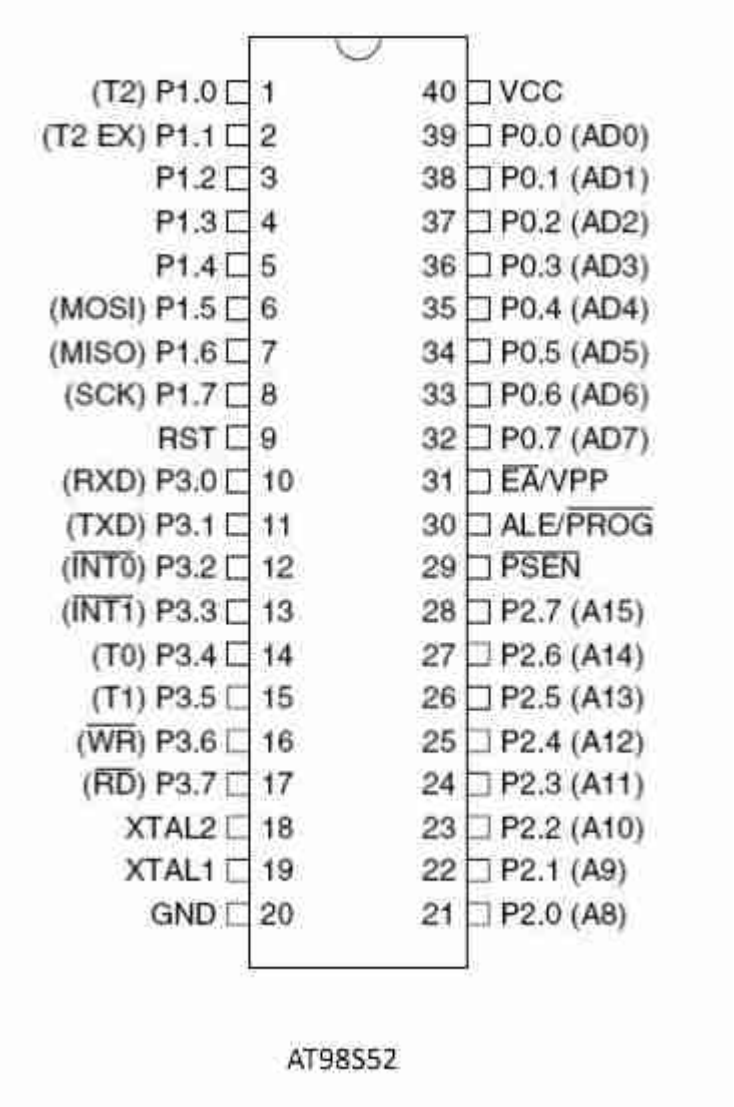
以下是 9 针串口的针脚功能定义：

表 5-1 串口针脚定义

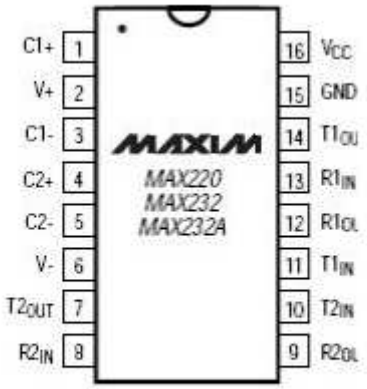
9针串口功能一览表

针脚	功能	针脚	功能
1	载波检测	6	数据准备完成
2	接收数据	7	发送请求
3	发送数据	8	发送清除
4	数据终端准备完成	9	振铃指示
5	信号地线		

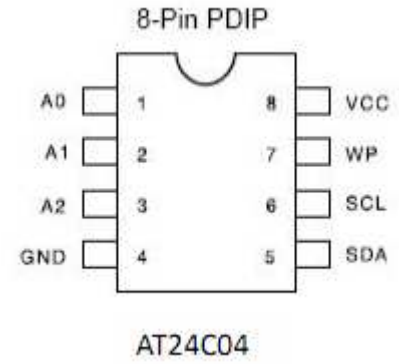
5.3 芯片资料参考



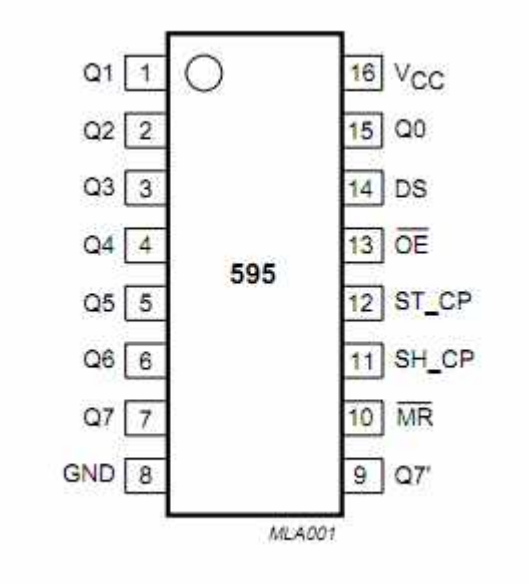
AT98S52



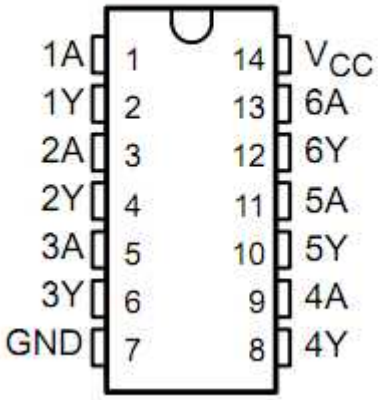
Max232



AT24C04



SN74S04 . . . D OR N PACKAGE
(TOP VIEW)



SN54HC125 . . . J OR W PACKAGE
SN74HC125 . . . D, DB, OR N PACKAGE
(TOP VIEW)