**南京信息职业技术学院**

毕业设计论文

作者 黄景琳 学号 11441P04

系部 电子信息学院

专业 物联网应用技术

题目 基于AVR单片机的电子万年历设计

指导教师 丁宁

评阅教师

完成时间： 2017年 4 月 30 日

毕业设计(论文)中文摘要

|  |
| --- |
| 题目：电子万年历的设计  摘要：电子万年历是生活当中相当多见的，也是使用广泛的电子产品之一。此次电子万年历的设计，采用ATmega32A单片机作为核心控制。由DS1302时钟芯片来提供年、月份、日期、礼拜等时间数据。用16\*64LED点阵屏显示。ATmega32是由美国的ATMEL公司推出的一款低耗能芯片。DS1302时钟芯片是由位于美国的DALLAS公司发行的低功耗实时时钟芯片。它可以对年、月、日、礼拜、时、分、秒，进行计时，此外还有闰年补偿等多种功能。并且DS1302的是有寿命长、误差小。数字显示采用的LED点阵屏，它可以满足人们对日常万年历的美观和显眼的要求。在软件方面主要包括主程序，独处日期时间程序两部分。所有程序编写完成后在Proteus上进行调试。  关键词：ATmega32A；电子万年历；DS1302 |

毕业设计(论文)外文摘要

**Title**: Design of electronic calendar

Abstract：Electronic calendar are quite common in our daily life and is one of the widely used electronic products. The design of the electronic calendar using ATmega32A controlled by Single Chip Microcomputer as the core. Provided by the DS1302 clock chip to year, month, date, week and time data. By 16\*64LED Matrix screen. ATMEL ATmega32 was done by the company launched a low-power chips. It can be for the year, month, day, week, hour, minute, second, timing, along with leap year compensation and other functions and DS1302 is a long life and Low error. The LED lattice screen digital display, it can satisfy people's requirement of daily calendar appearance and conspicuous. In terms of software includes main program and the date and time to be alone program. All programming is completed in Proteus for debugging.

Keywords: ATmega32A Design of electronic calendar DS1302

目录

[1 引言 1](#_Toc482352472)

[1.1 课题背景 1](#_Toc482352473)

[1.2 课题意义 1](#_Toc482352474)

[1.3 国内外现状及水平 2](#_Toc482352475)

[1.4 设计要求及任务 3](#_Toc482352476)

[2 硬件电路设计 3](#_Toc482352477)

[2.1 总体电路结构框图 3](#_Toc482352478)

[2.2 时钟芯片DS1302 4](#_Toc482352479)

[2.3 ATmega32A单片机 6](#_Toc482352480)

[2.3.1 主要特性 6](#_Toc482352481)

[2.3.2 工作模式 7](#_Toc482352482)

[2.3.3 单片机的内部结构 9](#_Toc482352483)

[2.3.4 单片机CPU结构 10](#_Toc482352484)

[2.3.5 存储器的结构 12](#_Toc482352485)

[2.4 显示电路 13](#_Toc482352486)

[2.5 单片机最小硬件电路 14](#_Toc482352487)

[2.5.1 复位电路 14](#_Toc482352488)

[2.5.2 电源电路 15](#_Toc482352489)

[2.5.3 振荡电路 15](#_Toc482352490)

[2.5.4 LED点阵屏显示电路 16](#_Toc482352491)

[3 系统软件设计与调试 16](#_Toc482352492)

[3.1 主程序设计 16](#_Toc482352493)

[3.2 读出日期时间程序 17](#_Toc482352494)

[3.3 系统的调试及性能分析 18](#_Toc482352495)

[3.4 PROTEUS软件简介 18](#_Toc482352496)

[3.4.1 PROTEUS的特点 19](#_Toc482352497)

[3.5 大功能模块 20](#_Toc482352498)

[3.6 Proteus简单应用 21](#_Toc482352499)

[3.7 系统调试与仿真结果 22](#_Toc482352500)

[结论 25](#_Toc482352501)

[致谢 25](#_Toc482352502)

[参考文献 25](#_Toc482352503)

[附录一 26](#_Toc482352504)

# 引言

随着第三次工业革命的爆发，电脑在我们平时生活中占有着重要地位，许多方面都需要用到计算机，单片机的诞生极大的促进了我们的生活。单片机具有不仅具有传统计算机（PC）的优点，还进一步改善了PC机的缺点，单片机的能耗很低、产品开发更加便利、单片机体积小、容易升级、外设丰富、能灵活运用等特点。因此，家用电子产品、机电控制、医学设备、汽车电子、网络和通信等领域中广泛使用。单片机的潜能被人们重视。万年历是人们日常生活中应用最广泛的时间工具，把单片机与万年历结合起来，是的万年历用起来更加方便与快捷。

万年历是我们生活中最常见的也是最重要的物理量之一，万年历是我们生活中一个十分重要的计时设备，尤其是计算机。本文主要介绍一种由ATmega32A单片机进行控制，以DS1302时钟芯片控制的时钟及 芯片控制的LED显示设备

## 课题背景

在当代繁忙的工作与生活中，时间与我们每一个人都有非常密切的关系，每个人都受到时间的影响，随着社会、科技的发展，人类得知时间，从观看太阳、沙漏到现在电子钟，不断研究、创新。为了在观测时间的同时，能够了解其它与人类密切相关的信息，比如温度、星期、日期等，电子万年历诞生了，它集时间、日期和星期功能于一身，具有读取方便、显示直观、功能多样、电路简洁等诸多优点，符合电子仪器仪表的发展趋势。伴随着电子技术的迅速发展，特别是随大规模集成电路出现，给人类生活带来了根本性的改变。由其是单片机技术的应用产品已经走进了千家万户。电子万年历的出现给人们的生活带来的诸多方便，作为一种附加功能，现在越来越广泛的被应用于各种电子产品中，具有广阔的市场前景。

## 课题意义

电子万年历作为电子类小产品不仅是市场上的宠儿，也是是单片机设计中一个很实用的题目。因为这个课题有很好的开放性和可发挥性，对制作者的要求比较高，不仅考察了对单片机的掌握能力更加强调了对单片机扩展的应用。而且要求设计的电子万年历在操作上力求简洁，功能上尽量齐全，显示界面也要出色。所以，电子万年历制作无论从实用目的，还是从培养能力的角度来看都是很有价值的毕业设计课题。本电子万年历的设计在硬件方面主要采用ATmega32A单片机作为主控核心，由DS1302时钟芯片提供时钟、LED点阵屏显示。ATmega32A是由ATMEL公司推出的AVR系列单片机中的一款。AVR系列单片机的耗能、效率、性能等方面都比以前推出的系列更强，优点更加明显。DS1302时钟芯片是美国DALLAS公司推出的低功耗实时时钟芯片，它可以对年、月、日、星期、时、分、秒，进行计时，而且DS1302的使用寿命长，误差小；数字显示是采用的LED点阵屏来显示，可以同时显示年、月、日、星期、时、分、秒等信息。在软件方面，主要包括读出日历和时间程序和显示程序等。所有程序编写完成后，在Proteus软件中进行调试，确定没有问题后，烧写到单片机上进行测试。最后在老师同学的帮助以及自己的努力下完成了此次电子万年历的设计。

## 国内外现状及水平

诸如定时自动报警、按时自动打铃、时间程序自动控制、定时广播、自动起闭路灯、定时开关烘箱、通断动力设备、甚至各种定时电气的自动启用等，但是所有这些，都是以钟表数字化为基础的。因此，研究万年历及扩大其应用，有着非常现实的意义。它可以对年、月、日、周日、时、分、秒,进行计时，对于数字电子万年历采用直观的数字显示，可以同时显示年、月、日、周日、时、分、秒等信息，还具有时间校准等功能。

综上所述此万年历具有读取方便、显示直观、功能多样、电路简洁、成本低廉等诸多优点，符合电子仪器仪表的发展趋势，具有广阔的市场前景。近些年我国也开始重视对电子万年历的开发与设计，让更多的电子时钟能够走进人民生活，跟多人能够应用到功能强大，精度高的电子时钟。但是仍然存在很多问题。

中国电子万年历产业发展出现的问题中，许多情况不容乐观，如产业结构不合理、产业集中于劳动力密集型产品；技术密集型产品明显落后于发达工业国家；生产要素决定性作用正在削弱；产业能源消耗大、产出率低、环境污染严重、对自然资源破坏力大；企业总体规模偏小、技术创新能力薄弱、管理水平落后等。  
 从什么角度分析中国电子万年历产业的发展状况？以什么方式评价中国电子万年历产业的发展程度？中国电子万年历产业的发展定位和前景是什么？中国电子万年历产业发展与当前经济热点问题关联度如何……诸如此类，都是电子万年历产业发展必须面对和解决的问题——中国电子万年历产业发展已到了岔口；中国电子万年历产业生产企业急需选择发展方向。  
 中国电子万年历产业发展研究报告阐述了世界电子万年历产业的发展历程，分析了中国电子万年历产业发展现状与差距，开创性地提出了“新型电子万年历产业” 及替代品产业概念，在此基础上，从四个维度即“以人为本”、“科技创新”、“环境友好”和“面向未来”准确地界定了“新型电子万年历产业” 及替代产品的内涵。根据“新型电子万年历产业” 及替代品的评价体系和量化指标体系，从全新的角度对中国电子万年历产业发展进行了推演和精准预测，在此基础上，对中国的行政区划和四大都市圈的电子万年历产业发展进行了全面的研究。

## 设计要求及任务

1. 设计任务及指标
2. 设计任务：

利用单片机和DS1302芯片，实现一个能精准供给年、月、日、礼拜、小时、分钟、秒钟等重要的时间数据。为现有的电子万年历系统设计提出一种新的设计方案，并需说明策划方案的构想依据、设计理念、系统工作原理、制作过程和系统运行流程图。

1. 技术指标：
2. 系统稳定性
3. 具有一定的自主调整时钟的能力
4. 使用芯片控制的LED显示屏
5. 设计方案

本次设计是一款实用便利的小型电子万年历，所采用的主要元器件有单片机ATmega32A、DS1302时钟模块， LED显示器，杜邦线若干。

# 硬件电路设计

## 总体电路结构框图

遵照生命周期法中第四阶段，系统设计中对功能的规定。可以确定系统主要由三个部分构成： ATmega32A主控制芯片、LED显示模块、DS1302时钟模块。万年历总体电路结构图如图 2‑1所示。

LED显示模块

DS1302时钟模块

ATmega32A

主控模块

图 2‑1 万年历总体电路结构图

## 时钟芯片DS1302

DS1302时钟芯片由位于美国的DALLAS公司发行的时钟芯片。DS1302可以对日常常见的时间单位进行同步计时，如：年、月、日等。而且DS1302的寿命长，差错很小。

1. 时钟芯片DS1302的工作原理

DS1302在每次进行读、写程序前都必须初始化，先把SCLK端置 “0”，接着把RST端置“1”，最后才给予SCLK脉冲；读/写时序如下图 2‑3所示。图 2‑2为DS1302的控制字，此控制字的位7必须置1，若为0则不能把对DS1302进行读写数据。对于位6，若对程序进行读/写时RAM=1，对时间进行读/写时，CK=0。位1至位5指操作单元的地址。位0是读/写操作位，进行读操作时，该位为1；该位为0则表示进行的是写操作。控制字节总是从最低位开始输入/输出的。表格 2‑1为DS1302的日历、时间寄存器内容：“CH”是时钟暂停标志位，当该位为1时，时钟振荡器停止，DS1302处于低功耗状态；当该位为0时，时钟开始运行。“WP”是写保护位，在任何的对时钟和RAM的写操作之前，WP必须为0。当“WP”为1时，写保护位防止对任一寄存器的写操作。

1. DS1302的控制字

DS1302的控制字如图 2‑2所示。控制字节的高有效位（位7）必须是逻辑1，如果它为0，则不能把数据写入DS1302中，位6如果0，则表示存取日历时钟数据，为1表示存取RAM数据；位5至位1指示操作单元的地址；最低有效位（位0）如为0表示要进行写操作，为1表示进行读操作，控制字节总是从最低位开始输出。

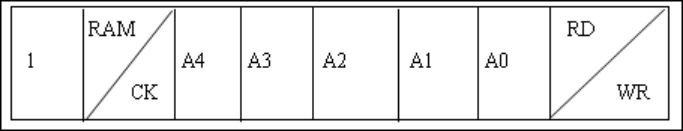


图 2‑2 DS1302的控制字

1. 数据输入输出

在控制指令字输入后的下一个SCLK时钟的上升沿时，数据被写入DS1302，数据输入从低位即位0开始。同样，在紧跟8位的控制指令字后的下一个SCLK脉冲的下降沿读出DS1302的数据，读出数据时从低位0位到高位7。如图 2‑3所示

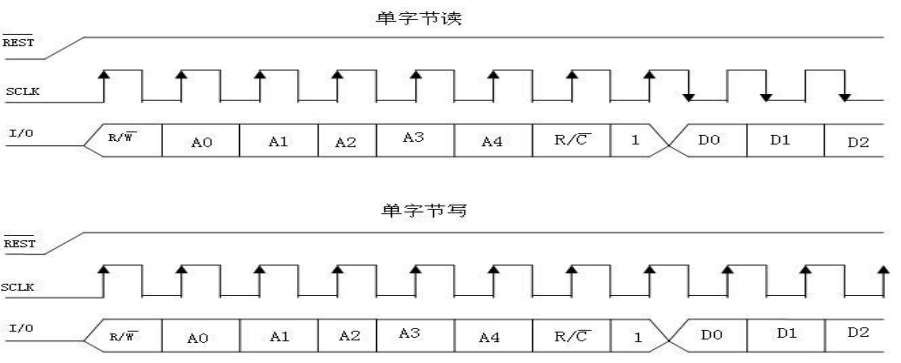


图 2‑3 DS1302的数据输入输出

1. DS1302的寄存器

DS1302有12个寄存器，其中有7个寄存器与日历、时钟相关，存放的数据位为BCD码形式,其日历、时间寄存器及其控制字见表格 2‑1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 写寄存器 | 读寄存器 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| 80H | 81H | CH | 10秒 | | | 秒 | | | |
| 82H | 83H |  | 10分 | | | 分 | | | |
| 84H | 85H | 12/ | 0 | 10 | 时 | 时 | | | |
|  |
| 86H | 87H | 0 | 0 | 10日 | | 日 | | | |
| 88H | 89H | 0 | 0 | 0 | 10月 | 月 | | | |
| 8AH | 8BH | 0 | 0 | 0 | 0 | 0 | 星期 | | |
| 8CH | 8DH | 10年 | | | | 年 | | | |
| 8EH | 8FH | WP | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

表格 2‑1日历、时间寄存器及其控制字

此外，DS1302 还有年份寄存器、控制寄存器、充电寄存器、时钟突发寄存器及与RAM相关的寄存器等。时钟突发寄存器可一次性顺序读写除充电寄存器外的所有寄存器内容。 DS1302与RAM相关的寄存器分为两类：一类是单个RAM单元，共31个，每个单元组态为一个8位的字节，其命令控制字为C0H～FDH，其中奇数为读操作，偶数为写操作；另一类为突发方式下的RAM寄存器，此方式下可一次性读写所有的RAM的31个字节，命令控制字为FEH(写)、FFH(读)。

## ATmega32A单片机

ATmega32A是由ATMEL公司推出的AVR系列单片机中的一款。AVR系列单片机的耗能、效率、性能等方面都比以前推出的系列更强，优点更加明显。ATmega32是以AVR增强的精简指令结构为核心的8位低耗能CMOS微控制器。由于它有一个优秀的指令集结构，使得单片机时钟周期运行时间很快。ATmega32的数据吞吐量最快可达1MIPS/MHz。从而根本上解决了过去能耗与速度之间的冲突。传统的单片机处理速度快必定能耗很高，这与目前行业发展要求是相违背的。人们需要的是能耗更低、效率更高、体积更小、物美价廉的产品。

### 主要特性

1. 拥有优秀的Flash程序存储器，可以频繁擦写，为了方便产物的调试与开发、生产与维护，还提倡使用ISP和IAP。
2. 具有高效率、低耗能和六种休眠功能。
3. 不仅有超功能精简指令，还拥有32个通用工作寄存器。这可以解决简略累加器数据处理酿成的瓶颈现象。
4. 有34个中断源，不同中断向量的入口地址不同，可快速响应中断。
5. 工作电压：在4.5V~5.5V之间。工作频率范围在0-16MHz。
6. 具有数据存储空间位32K字节和芯片中集成2048B SRAM 。
7. 单片机内部ISP Flash 可以通过三种方式接入。第一种是ISP 串行接口。第二种用通用编程器进行编程。第三种是通过运转在AVR 芯片之中的Boot进行编程。
8. 拥有3个具有比较模式定时器/计数器。
9. 有3路外部中断。可以由下降沿、上升沿或者低电平触发电路。通过电平办法触发中断可以将MCU 从掉电模式叫醒。但是要确保电平保持一定的时间，以降低MCU 对噪声的敏感程度。
10. 具有EEPROM功能和看门狗功能。工作温度在-55°C 至 +125°C之间。
11. USART是全双工操作。支持多种类型的数据位和1个或2个停留位，三个独立中断与UART兼容。
12. ATmega32A拥有三种封装方式。分别是双列直插式封装方形封装和贴片形式封装。

### 工作模式

ATmega32A引脚如图 2‑4所示。

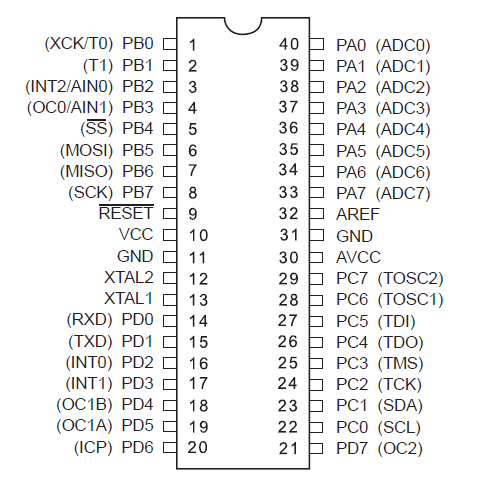


图 2‑4 ATmega32A引脚图

ATmega32A的主要引脚功能

1. 主电源引脚：

VCC（5脚）：接+5V电源正极；

GND（20脚）：接+5V电源接地端。

1. 外接晶体引脚

XTAL1（13脚）：反向震荡放大器与片内时钟操作电路的输入端；

XTAL2（12脚）：反向震荡放大器的输出端。

1. 控制信号引脚

RESET（9脚）：复位输入引脚。持续时间超过最小门限时间的低电平将引起系统复位。

AREF（32脚）：A/D的模拟基准输入引脚。

AVCC（30脚）：是端口A与A/D转换器的电源。

1. 输入/输出

PA口（33脚-40脚）：PA0-PA7统称为PA口。端口A第二功能是作为A/D转换器的模拟输入端。

PB口（1脚-8脚）：PB0-PB7统称为PB口。端口B为8位双向I/O

端口A第二功能是作为模拟信号/数字信号转换器的模拟输入端。

端口B有8位双向输入/输出口。用来传输地址或控制数据。

PB口还具有第二功能，详见表格 2‑2。

|  |  |
| --- | --- |
| 引脚 | 第二功能 |
| PB0 | XCK（USART外部时钟输入/输出）  T0（T/C0外部计数器输入） |
| PB1 | T1（T/C1外部计数器输入） |
| PB2 | AIN0（模拟比较负输入）  INT2（外部中断2输出） |
| PB3 | AIN1(模拟比较负输入端)  OC0（T/C0输出比较匹配输出） |
| PB4 | （SPI从机选择引脚） |
| PB5 | MOSI（SPI总线的主机输出/从机输入信号） |
| PB6 | MISO(SPI总线的主机输入/从机输出信号) |
| PB7 | SCK(SPI总线的串行时钟) |

表格 2‑2 PB口第二功能

端口C有8位双向输入/输出口。用来传输地址或控制数据。还有可编程的内部上拉电阻，详见表格 2‑3。

|  |  |
| --- | --- |
| 端口引脚 | 第二功能 |
| PC 7 | TOCS2（定时振荡器引脚2） |
| PC 6 | TOCS1（定时振荡器引脚1） |
| PC 5 | TDI（JTAG测试数据输入） |
| PC 4 | TDO（JTAG测试数据输出） |
| PC 3 | TMS（JTAG测试数据选择） |
| PC 2 | TCK（JTAG测试时钟） |
| PC 1 | SDA（两线串行总线数据输入/输出线） |
| PC 0 | SCL（两线串行总线时钟线） |

表格 2‑3 PC口第二功能

端口D有8位双向的I/O口。用来传输地址或控制数据。具有可编程的内部上拉电阻，详见表格 2‑4

|  |  |
| --- | --- |
| 端口引脚 | 第二功能 |
| PD 7 | OC2（T/C2输出比较匹配输出） |
| PD 6 | ICP1（T/C1输入捕捉引脚） |
| PD 5 | OC1A（T/C1输出比较A匹配输出） |
| PD 4 | OC1B（T/C1输出比较B匹配输出） |
| PD 3 | INT1（外部中断1的输入） |
| PD 2 | INT0（外部中断1的输出） |
| PD 1 | TXD（USART输出引脚） |
| PD 0 | RXD（USART输入引脚） |

表格 2‑4 PD口第二功能

### 单片机的内部结构

1. 一个32位中央处理器
2. 内部有三个定时器/计数器，分别是2个8位定时器/计数器和1个16位定时器/计数器，一个是具有PMW功能的8位定时器/计数器0（T/C0），一个是16位的通用定时器/计数器1（T/C1）,还有一个是具有PMW功能和异步操作的定时器/计数器（T/C2）。
3. 3个逻辑存储空间
4. 32K系统内可编程Flash。它的擦写寿命高达一万次。
5. 1024B的EEPROM，擦写寿命为十万次。
6. 2048B的片内SRAM。
7. 具备单独锁定位的可选的引导程序代码区，通过芯片上的引导程序完成系统内编程，切实的完成同时读与的写操作。
8. 通过对锁定位的编程达到对用户程序的加密
9. 4组双向可按位寻址的I/O端口。
10. 34个中断源。
11. 片内有振荡器和时钟电路。

### 单片机CPU结构

ATmega32A单片机的CPU结构如下图 2‑5。

间接寻址

直接寻址

程序计数器

状态和控制

程序存储器

32个8位通用寄存器

ALU

数据存储器

EEPRON

I/O线

指令寄存器

指令译码器

控制线

中断单元

SPI单元

看门狗

定时器

模拟比较器

I/O Module 1

I/O Module 2

I/O Module 3

图 2‑5 ATmega32A单片机的CPU结构

1. 运算器的主要功能
   1. 算术运算和逻辑运算：可以对4位二进制，8位二进制等数据进行操作。
      1. 可以完成简单的算术运算
      2. 可以处理基本的逻辑运算
   2. 内部容纳布尔处理器，应用于实行位运算的操作。
      1. 置位、取反、清零等操作。
   3. 可以进行简单的硬件乘法操作。一次硬件乘法操作需要2个时钟周期。
2. 程序计数器PC

程序计数器主要是用来应用在存储即将要执行的指令的地址，总共14位，可以对16K的程序存储器实行直接寻址。引导程序区和相关的软件安全锁，可以边读边写，有自我编程能力。常数可以直接存储在程序存储地址空间当中。可以认为，当程序运转时，运行到哪里程序计数器就对着哪里。把当前所指令的地址中的指令送到指令寄存器中，这是PC的取指令过程。当取完一次指令之后，指令寄存器PC会自己主动的加一，将当前地址对着下一个即将要取指令的地址。除非重启，否则指令寄存器会无限加一下去，他始终跟随着程序变化。众所周知的是，用户程序是单独存放在ROM之中，我们想要执行用户程序的话就要从ROM中一个一个字节的读出来，然后放在CPU中去执行，然而ROM执行那一条是由PC来指引的。

程序计数器（PC）拥有自主加1功能，也就是说，从存储器读出一个字节的执行代码后，程序计数器（PC）就会自己给自己加1（将指针对着下一个存储单元的地址）。

1. 指令寄存器IR

指令存储器（IR）主要是用来保存将要执行的指令代码。CPU执行指令的整个过程分为以下四步。第一步，由程序存储器（ROM）中读取指令代码的地址。第二步，找到那个地址之后将地址里存储的代码送进道指令寄存器当中，第三步，经过译码器译码进行译码。第四步，靠定时与控制电路发出和它相对应的控制信号，进而完成指令相对应的操作。

1. 指令译码器ID

指令译码器（ID）主要是将送进IR中的指令经译码器施行译码。译码器就是把指令改变成，履行此指令所需要的电信号。当指令送入IR后，译码器主要履行对该指令经行译码的职责。CPU控制电路再依据译码器输出的信号形成实施指令所需要的一系列信号，以致单片机精确的施行程序所需要的各种操纵。

1. 地址寄存器AR（14位）

地址寄存器（AR）主要是用来寄存即将要寻址的外部存储器单元的地址信息。指令是由操作码和操作数两部分组成的。操作码是存储单元所在的地址编码，由PC产生。操作数规定了指令中进行运算的量。操作数可以存放2种数据。第一种是操作数本身要运算的数据。第二种是存放操作数所在的地址。由图 2‑5可知AR通过AB总线与外部存储器相连的。

1. 数据寄存器DR

数据寄存器（DR）用于存储放入External memory 或者Input /Output port的数据。由上图2-3可以看到DR与DB直接连接。

1. 程序状态字PSW

用来记录运算过程中单片机的状态，查看程序是否溢出或需要进位等等。

1. 时序部件

时序零件是产生单片机在施行指令进程中的时序信号。单片机在工作流程中是一个指令周期接着一个指令周期进行的，在一个指令周期内是一个节拍跟着一个节拍的工作。在各种不同指令的不同机器周期的不同节拍中产生什么控制信号，是由指令规定的。时序部件通常由脉冲源、节拍电位发生器和启停逻辑三个部分组成。

### 存储器的结构

一般PC机中都是将程序和数据共同存放在一个存储空间，这种结构称之为普林斯顿结构。然而大多数单片机，为了数据保密和防冲突，一般是将程序存储器和数据存储器完全分离寄存的，分别存放在RAM和ROM中，单独编址并且分别寻址，使得他们之间相互不会冲突，这种构造称为哈佛结构。ATmega32A的内部存储器结构是哈佛结构。ATmega32A除了上述两个存储器外，还有EEPROM以保存数据。ATmega32A的存储器都是线性的平面结构。

1. 可编程的 Flash ROM

ATmega32A具有32KB在线编程Flash，用来存放程序指令代码。用户程序的安全性要根据Flash程序存储器的两个区，即引导（Boot）程序区和应用程序区分开来考虑。Flash ROM可擦写10000次，ATmega32A的程序计数器（PC）为14位，因此可寻址16KB的程序存储器空间，常数可以保存于整个程序存储器地址空间。

1. SRAM数据存储器

2144个数据存储器包括了存储器文件、I/O存储器和内部数据SRAM。起始的96个地址为寄存器文件与64个I/O存储器，接着是2048B的内部数据SRAM。

数据寻址方式分有5种，分别是直接寻址、带偏移量的间接寻址、间接寻址、带预减量的间接寻址和带后增量的间接寻址。寄存器文件中的寄存器R26~R31为间接寻址的指针寄存器，直接寻址范围可达整个数据区域。带偏移量的间接寻址模式能够讯知道有寄存器Y和Z给定的基址附近的63个地址。在自动预减和后加的间接寻址模式中，寄存器X，Y和Z自动增加或减少。ATmega32A的全部32个通用寄存器、64个I/O寄存器及内部数据SRAM可以通过所有上述的寻址方式进行访问。

1. EEPROM数据存储器

ATmega32A包含了1024B的EEPROM数据存储器。它是作为一个独立的数据空间存在的，可以按字节读写。EEPROM的寿命至少为100000次擦除周期。EEPROM的访问有地址寄存器、数据寄存器决定。

## 显示电路

显示电路我们选用了LED点阵屏动态扫描的方式来显示。一个8\*8LED点阵屏是由64个发光二极管组成的，如图 2‑7。每个发光二极管放置在行与列相交的点上，当某一行置高电平，某一列置低电平时，相对应的发光二极管就会亮。

所谓动态扫描的显示方式，是指在一个时钟信号或中断发生时扫描并更新LED屏幕上所显示的数据。每次数据更新时，LED点阵屏行会导入扫描行的选通信号。当列上的发光二极管接收到有一个负向脉冲选通信号时，列线上的数据位在相应的位置会置“1”且发光二极管导通点亮。依照逐行扫描的方式，使得数据不断的输入并扫描。一个中断时间内只扫描一种数据。因为扫描的速度很快，当频率为24帧/秒或25帧/秒时，达到人体视觉余晖效应。让人感觉在这个时间段里面只有一个画面。其实此时的发光二极管时在不断闪烁的。直到下一个中断时间更新数据，才可以扫描出另一个图像。由于动态显示的亮度比静态显示亮度稍弱，所以挑选限流电阻应该小于静态显示电路电阻。

动态显示方法下，LED点阵所有端口根据端口定义插入相对应的I/O端口中不论需要多少个LED点阵屏只需要一块CPU就可以控制，后面只要调整输出信号，让LED点阵不断地并联，点阵最大点数小于不超过RAM寻址最大值即可。8\*8LED点阵如图 2‑7，LED点阵正面（图 2‑6）。

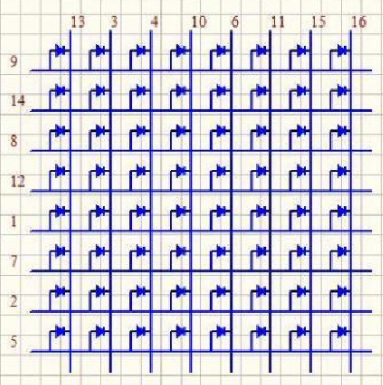
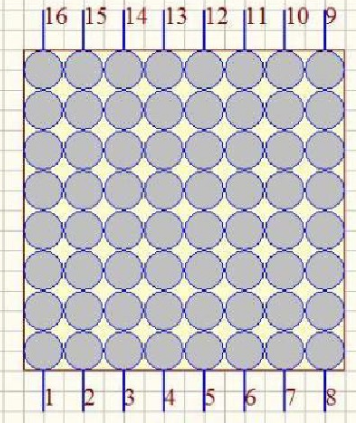
 

图 2‑6 LED点阵 电路图 图 2‑7 LED点阵 正面

由于动态LED点阵屏的以上优点，在单片机控制系统中还有显示生活中都经常使用。像路边的LED广告，校园里的LED点阵校园温馨提示，都是使用LED点阵屏基于单片机的控制所制造的。LED点阵许多生产商已经把多个单位的LED点阵集成在一起形成LED点阵屏，如图 2‑8。

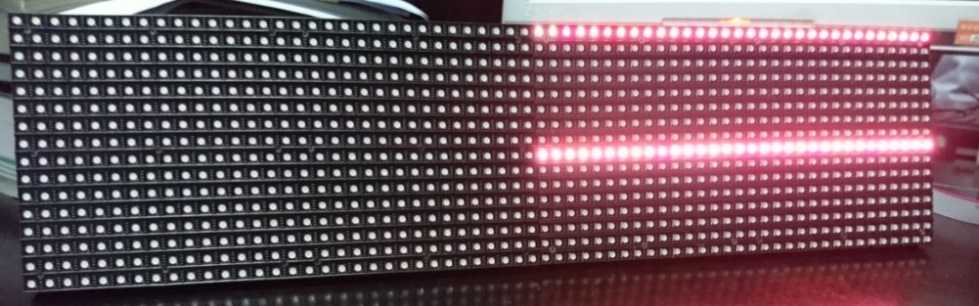


图 2‑8 LED点阵屏实物

## 单片机最小硬件电路

MCU的正常工作复位电路和时钟电路是必不可少的，因此单片机最小运行环境至少应该由单片机C、复位电路和时钟电路，如图 2‑9所示。

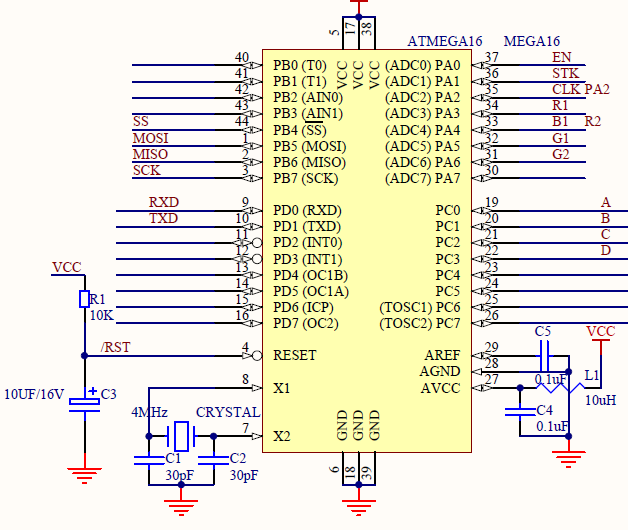


图 2‑9 单片机最小系统

### 复位电路

用户在任何时候只要涉及单片机的设计。就要充分考虑到单片机种复位电路的设计。然而复位电路的优劣程度，往往直接决定了整个系统的可靠性。许多学生在设计完成整个作品后，在实验室调试也成功了。但是到现场却出现了混乱，卡死等情况。这主要是由于单片机的复位电路设计欠佳导致的。本次电路设计选择可靠的手动复位，单片机复位方式如图 2‑10所示。

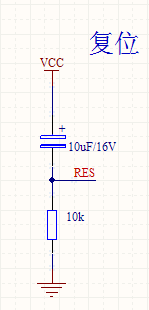


图 2‑10 复位电路

### 电源电路

由电池或USB接口供给5V电压。如图 2‑11所示，这种电源电路可以给单片机提供一个稳定的电压，让整个系统可以稳定工作。

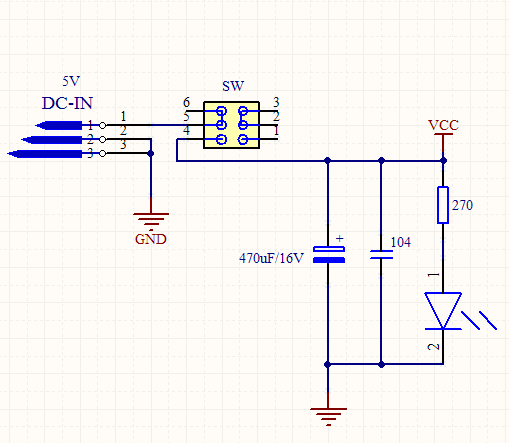


图 2‑11 电源电路

### 振荡电路

对于一个高效能稳当的系统策划，选择晶振是至关重要的。在震荡电路中，既不能过激励也不能欠激励。因此，晶振的选择基本上要考虑以下几点。第一，谐振频率与负载电容的大小。第二，激励功率以及温度的影响。第三，长期稳定性。

在本设计中采用了12MHz的晶振。单片机震荡电路如图 2‑12所示。

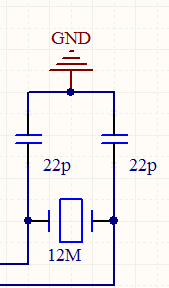


图 2‑12 振荡电路

### LED点阵屏显示电路

LED点阵屏在日常生活中使用广泛，是一种常见的电子显示设备。此次选用的是32\*64 LED点阵屏，LED点阵模块如图 2‑13所示。

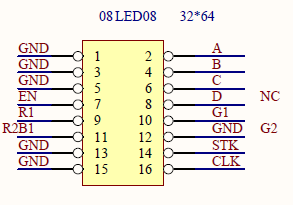


图 2‑13 LED点阵屏显示电路

# 系统软件设计与调试

系统程序主要包括主程序，读出日期时间子程序。

## 主程序设计

系统控制程序采用C语言编写，按照模块化设计的思路来设计。首先分析程序所要实现的功能，程序要实现实时时钟信息的获取、LED显示屏的静态显示与动态显示三大功能。实时时钟数据由DS1302获取后，交给主程序处理后再通过显示控制程序选择不同的显示方式进行显示。

N

Y

Y

开始

显示

判断DS1302是否设置过时间

初始化时间

N

调用读出日期和时间子程序

是否到1s？

系统初始化

图 3‑1 主程序流程图

主程序的主要功能是负责实践的实时显示、读出并处理DS1302时钟芯片的数据及指令，每1s刷新一次当前时间。这样可以直到当前的实时时间，流程图详见图 3‑1。

## 读出日期时间程序

读出日期时间子程序的主要功能是读出RAM中的31个字节，当时钟到1秒是主程序回访问读出日期时间程序，其程序流程图如图 3‑2所示:

否

是

否

是

开始

变量初始化

DS1302不写保护

复位产生一个高电平

写DS1302地址

延时一段时间

向该地址写数据

地址增加

数据是否写入

复位产生一个高电平

写DS1302地址

延时一段时间

将该地址的数据读出

地址增加

数据是否读入

显示数据

图 3‑2 读出日期时间流程图

## 系统的调试及性能分析

硬件调试拔取PROTEUS软件绘制原理图。利用软件自带的调试功能进行仿真。通过软件调试我们可以对硬件功能的正确性进行检查和验证。Proteus可以很直观的显示并告诉我们错误在哪里。我们分别对系统软件的两个部分进行调试。

## PROTEUS软件简介

Proteus是位于英国的Labcenter electronics公司推出的一款EDA软件。现在由广州风标电子技术有限公司负责担任中国总代理。它拥有传统EDA软件的绘图能力，还能对单片机及外围设备进行仿真。因此，他是目前行业中最喜欢的电子设计出版工具自动化工具软件之一，也是最好的仿真单片机及外围器件的工具。虽然中国推行此刻仍处在初级阶段，但已广受单片机设计单片机玩家、投身与单片机设计行业的员工、高等教育院校中担任单片机教学的教师以及致力于单片机开发及应用的科研工作者的喜爱。Proteus是当今世界上最富盛名的EDA软件之一。从原理图的设计、PCB板的布线及设计到代码调试与单片机与外围电路的仿真都是一个软件完成的。真正达到产物从抽象概念构想到产品实物出生的完美设计。是当下最富盛名的EDA软件。将电路仿真软件、PCB设计软件和虚拟模型仿真融合在一起的软件。这三个软件在开发重要过程起到重大作用。将他们融合在一起之后可以解决很多软件方面问题。用户不用再担心软件匹配方面的问题。企业也可以节约很多培训成本。Proteus的处理器模型支持常见的单片机型号如：8051、MSP430等，新版的Proteus Pro 7 增加了DSP、IAR等一系列新型处理器。此外，还增加了ADC/DCA等期间的增强的嵌入式外设库，还有ENC28J60和RTL8019AS以太网控制器模型。另外新版的PBC制作海推出了3D可视化工具，不需要再担心做出来高度外形不好而需要重新设计布局，可视化3D工具使用户可以花更少的时间和金钱去做更优秀的产品。

### PROTEUS的特点

1. 实现了单片机仿真和集成电路仿真相融合。拥有数字/模拟电路仿真、各种常见的MUC及外围电路（如LED、RAM、ROM、键盘、模拟/数字转换器……）组成的仿真系统。
2. 供给了多种虚拟化仪器。如示波器、信号发生器、电压表、电流表等，使得调试更加方便快捷，真正做到显示场景的模拟仿真。
3. 不仅提供基本的硬件调试功能，还支持第三方软件的编译和调试如Keil等软件。
4. 有超齐全的电路原理图绘图功能。Proteus和其他同类软件不同。它不仅可以仿真MUC中芯片内部的工作情况，也可以仿真MUC外部的电路，不论那个器件有没有参加其他电路的工作情况。所以我们在仿真和程序调试的时候，不用担心写的语句的时候对MCU内部的影响。而是直接从宏观的角度看程序运行时是否能满足我们所定下的需求。这样我们就可以节省更多的时间去做别的事情，对仿真实验也有好处。从实际操作来说，避免了调试实验和工程实际常常脱离的矛盾现象。然而，当硬件调试完成之后。我们就可以利用Proteus 的ARES软件，就可以简单的获得其PCB图。为用户后期其他的创造给予了方便。

## 大功能模块

1. 智能原理图设计（ISIS）

丰富的器件库，超过两万种元器件，可以跟间接的船艰辛的元器件。

智能的器件搜索，即使忘记全名也可以通过模糊搜索迅速定位。

现代化的连线功能：自动连线功能，可以减少工程师重复大量单一耗时的工作，真正的为工程节约了绘图时间。

支持总线结构：利用总线器件可以容电路设计更加直观，方便其他工程师对项目的理解

超高清高品质图纸：可以生成各类型的图纸，供给不同的软件使用。满足用户在不同领域上对图纸的需求。

1. 健全的电路仿真功能（Prospice）

Pro SPICE 混合仿真：基于工业标准SPICE3F5。实现模拟/数字电路的混合仿真。超齐全的种类繁多的仿真元器件，可以使用软件自带的模型库或使用厂家给予的SPICE文件自己设计，Labcenter 也在不断地发布新兴的仿真元器件，还支持第三方发布的仿真器件。

具有丰富的的激励源：拥有丰富的激励源和各种各样的信号，满足使用者的所有需求。例如：正弦、分段性脉冲等。

具有丰富的虚拟仪器：13种常见的虚拟仪器，超高还原度的操纵面版。如：电流表、电压表等。

1. 独立的单片机共同仿真功能（VSM）

支持现在市面流通的MCU类型，如ARM7、AVR、MPS430等，MUC类型随着软件更新还会陆续增添，如DSP处理器。

支持常用外设模型：如LED点阵、LED七段显示模块、键盘/按键、直流/异步/步进/伺服电机等，其中COMPIM还可以是电路通过PC机串口和外部电路实现双向异步串行通信。

实时仿真：支持中断仿真、SPI/I2C仿真、ADC仿真等。

编译与调试：支持单片机汇编语言的编辑/编译。源码级仿真。内附AVR、PIC的汇编编译器，也可以使用第三方集成的编译环境，惊醒高级语言的代码及仿真与调试。

1. 便利的PCB设计平台

原理图到PCB的快速通道：原理图设计完成后，一键就可以进ARES的PCB设计环境，实现从产品的设想到产品的制造的全部设计。

优秀的自动布线/布局功能：支持器件的人工/自动布局，支持无网格自动/人工布线，支持引脚/门的交换使得PCB设计更为合理。

完整的PCB设计功能：最多16层铜箔层，2个丝印层，4个机械层，灵巧的不限策略给用户设置，自动的设计规则检查以及3D可视化浏览使得用户更方便，更直观的完成PCB的设计。

支持多样的输出格式：有丰富的输出文件格式，包括Gerber文件的导入与导出，方便地与其他PCB设计工具呼唤和PCB板的设计与加工。

## Proteus简单应用

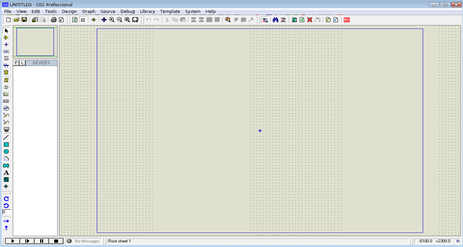


图 3‑3 Proteus 界面图

1. 绘制原理图：Proteus界面如图 3‑3所示。绘制原理图要在原理图编辑窗口中的蓝色框框中绘制。Proteus的原理图编辑和我们平常使用的软件有很大的不同。它是左键拖放，右键选择、双击删除。正好和我们平时使用习惯相反。右键拖出一片区域，可以选择多个元件。不论编辑、删除还是修改都遵循先右键再左键。连线操作：左键连线，右键删除。鼠标滚动可以对原理图进行放大或缩小。
2. 定制自己的电子元件有三种方式：
3. 使用PROTEUS VSM SDK开发仿真模型，并制作元件。
4. 在已有的元件基础上进行改造。
5. 利用别人已画好的原件，通过网上下载一些新原件并把它们增添到自己的元件库中。
6. Sub-Circuits应用：用子电路把部分电路封装起来，这样可以简略原理图窗口空间。

## 系统调试与仿真结果

从软件的元件库中找到需要的元器件，并且完成仿真所需要的电路，图 3‑4所示。

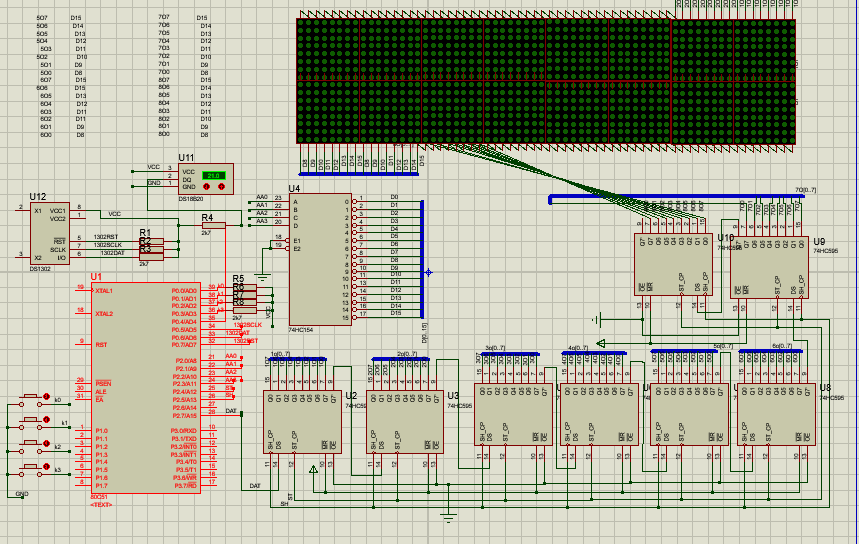


图 3‑4 Proteus仿真电路图

双击ATMEGA32选择仿真所需要的HEX文件，过程如图 3‑5所示。

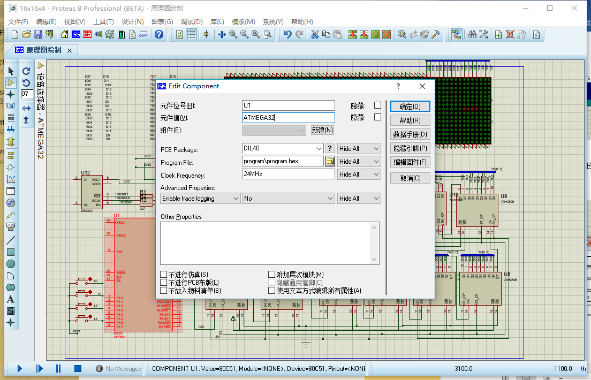


图 3‑5 引用需要的HEX文件

然后点击左下角的运行仿真键开始仿真。图 3‑6为当前年份的仿真结果，图 3‑7为当前日期显示的仿真结果，图 3‑8为当前时间的仿真结果，图 3‑9为当前星期的仿真结果。

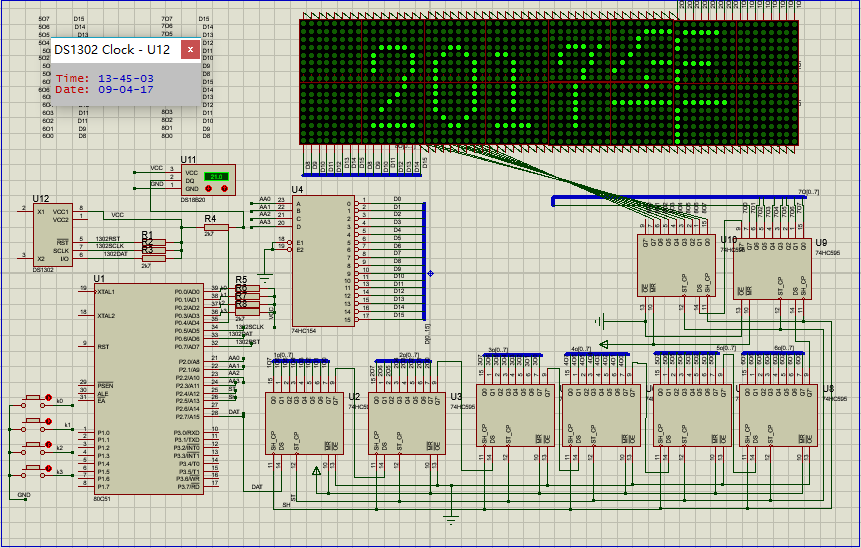


图 3‑6 显示当前年份的仿真效果

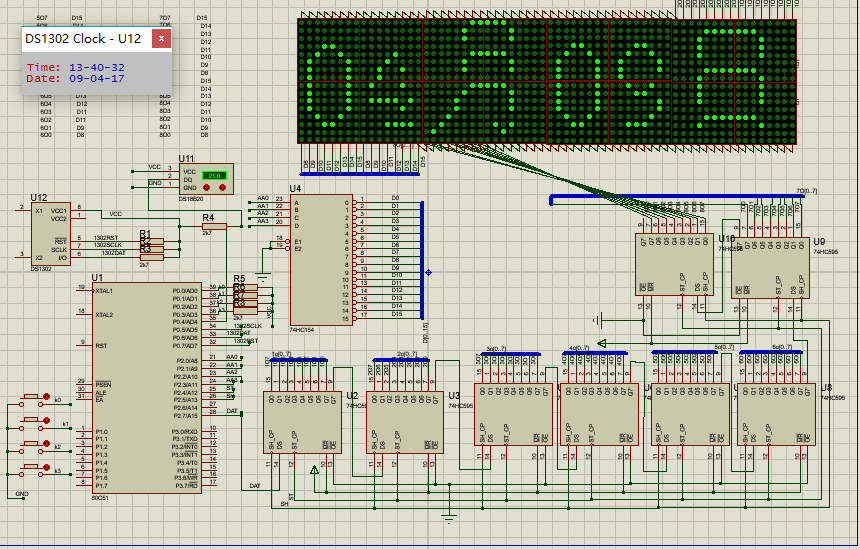


图 3‑7 显示日期的仿真效果

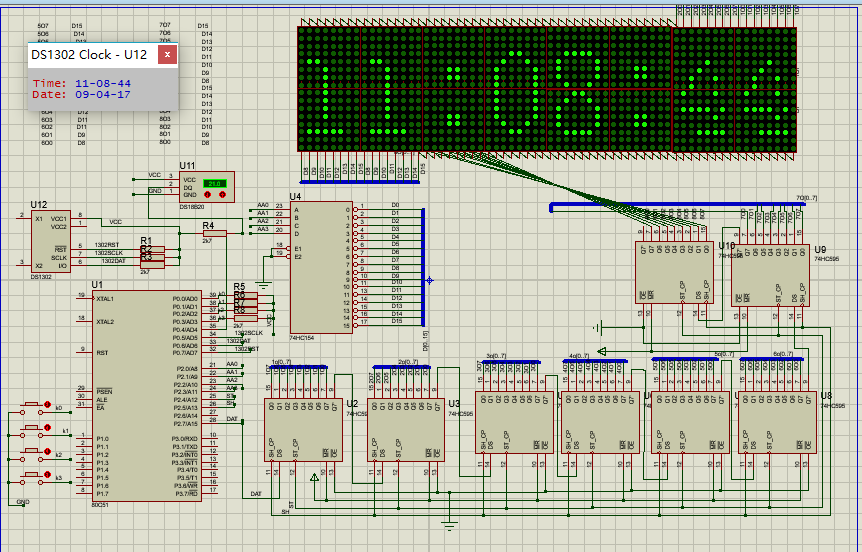


图 3‑8 显示前时间的仿真效果

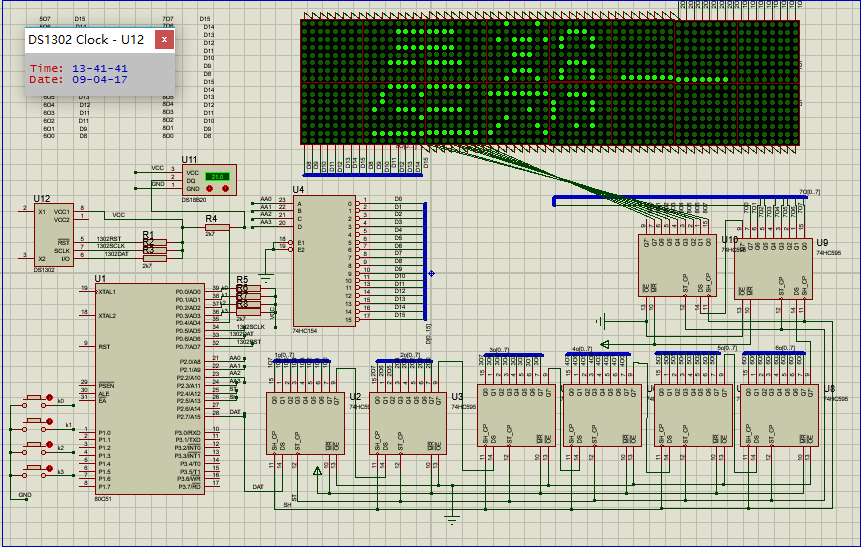


图 3‑9 显示星期的仿真效果

结论

通过本次设计令我稳固课本上的知识，真正做到了学以实用。这是我第二次自己动手设计电路，通过系统仿真软件Proteus，使我更深入了解了单片机的设计制作过程，一开始AVR studio由于系统使用win10的原因导致初期平台构建的时候就出现了许多问题，而且因为win10系统比较新网络上很难找到相关的解决方法，现在使用依旧会出现一些问题，但不影响软件的正常使用，在整个设计中最为艰苦的是软件部分，尤其是编程部分，由于C语言只学习了基础知识，有很多高级应用，像指针之类的并不太会用，导致花费了大量时间在网络上学习相关知识，然后单片机的使用也不是很灵巧出现问体时也需要在网络上寻找了大量的资料，在老师、同学还有网络的帮助下，经过自己修改之后，依旧有许多功能不得以实现，如显示的翻页功能，公历转农历功能等。由于对Proteus的使用并不是十分熟悉，大部分原件的英文名不清楚，导致花费了大量的时间在网络上查询，今后应该在英语方面多家努力。

致谢

本次毕业设计首先要感谢我的指导老师丁宁老师，她在硬件设计方面给了我许多的指导，其次要感谢聂佰玲老师，她在我软件设计方面给予了重要的帮助与指导，让我对单片机内部的编程以及C语言有了更深入的了解。谢谢丁宁老师和聂佰玲老师给予我的帮助。

参考文献

[1] 曾若渊.MCS-51定时器/计数器在出租车计价器中的应用.现代测量与实验室管理[J].2005(01):24-25.

[2]杜峰袁显举,姚立影,赵永先.基于Proteus的多路信号监测系统仿真[J].绵阳师范学院学报.2013(02):39-44.

[3]耿德新，杨正忠.AVR单片机应用开发指南及实例精解[Z].北京.中国电力出版社,2008.

[4]黎锦钰.基于单片机的LED点阵时钟的制作[J].信息通信.2014(09):48-49.

[5]梁力中.基于个域网的无线温湿度传感器设计及医学实验室应用[J].现代电子技术.2013(11):118-119+123.

[6]柳秀山.EDA教学与现代化电子设计能力的培养[J].广东技术师范学院学报.2004(04):95-97.

[7]申万增,燕宏斌,高大庆,周忠祖,张显来.扫描磁铁电源的仿真与实现[J].核技术.2014(02):47-50.

[8]王立.电子技术实验教学中创新能力培养的探索[J].长春师范学院学报. 2005(05):136-138.

[9]王轩朱东起,姜新建.三相四线制下串联混合型滤波器的研究[J].电力系统自动化.2004(03):50-53.

[10]吴志超孟现岭,赵地,鞠阳.光伏发电中基于拉格朗日插值法的最大功率点跟踪[J].华北水利水电大学学报.自然科学版,2014(05):82-84.

## 附录一

#include <avr/io.h>

#include "main.h"

#define F\_CPU 4000000UL // 4 MHz

#include <avr/interrupt.h>

#include "font\_library.h"

#include <util/delay.h>

unsigned char T0\_Counter = 0,Flag=0,T2\_Counter = 0,Flag\_2=0; //定时器参数初始化 Flag控制时间 Flag\_2控制日期

unsigned char h = 8,m = 30,s = 10,d = 12,moth = 5;

unsigned int y=2017;

void LEDconfig()//LED初始化

{

DDR\_LINE |= ((1<<RowA)|(1<<RowB)|(1<<RowC));

PORT\_LINE |= ((1<<RowA)|(1<<RowB)|(1<<RowC));

DDR\_CTL |= ((1<<EN)|(1<<STK)|(1<<CLK));

PORT\_CTL |= ((1<<EN)|(1<<STK)|(1<<CLK));

DDR\_DATA1 |= ((1<<R1)|(1<<G1)|(1<<B1));

DDR\_DATA2 |= ((1<<R2)|(1<<G2)|(1<<B2));

PORT\_DATA1 |= ((1<<G1)|(1<<R1)|(1<<B1)); //0使能 1关断 控制上半层0~7

PORT\_DATA2 |= ((1<<G2)|(1<<R2)|(1<<B2)); //0使能 1关断 控制下半层8~16

}

void Time\_counter() //控制显示时间

{

s+=1;

if(s>=60)

{ m+=1;

s=0;

if(m>=60)

{

h+=1;

m=0;

if(h>=24)

{

d += 1;

h=0;

if(moth == ( 1 || 3 || 5 || 7 || 8 || 10 || 12 ))

{ if(d>=31)

{

moth+=1;

d=1;

}

}

else if(moth == (4 || 6 || 9 || 11))

{

if(d>=30)

{

moth+=1;

d=1;

}

}

else if(moth == 2)

{

if((y%4 == 0) && (y%100!=0)||(y%400==0))

{

if(d>=29)

{

moth+=1;

d=1;

}

else

{

moth+=1;

d=1;

}

}

}

if(moth>=12)

{

y += 1;

moth = 1;

}

}

}

}

}

void LineWrite(unsigned char r)

{ //输出行线状态ABCD （A低,D高)

PORT\_LINE &=0xF0;

PORT\_LINE |=r;

}

void SendData(unsigned char dat\_up,unsigned char dat\_down)

{

unsigned char i = 0;

for(i = 0; i < 8; i++)

{

if(dat\_up & 0x01)

{

PORT\_DATA1 |=(1<<R1);

PORT\_DATA1 |=(1<<G1);

PORT\_DATA1 |=(1<<B1);

}

else

{

PORT\_DATA1 &=~(1<<R1);

PORT\_DATA1 &=~(1<<G1);

PORT\_DATA1 &=~(1<<B1);

}

if(dat\_down & 0x01)

{

PORT\_DATA2 |=(1<<R2);

PORT\_DATA2 |=(1<<G2);

PORT\_DATA2 |=(1<<B2);

}

else

{

PORT\_DATA2 &=~(1<<R2);

PORT\_DATA2 &=~(1<<G2);

PORT\_DATA2 &=~(1<<B2);

}

CLK\_1;

\_delay\_us(1/24);

dat\_up >>= 1;

dat\_down >>= 1;

CLK\_0;

}

}

void display()

{

unsigned char x = 0;

unsigned char y = 0;

for(y = 0;y < 8; y++){

PORT\_CTL |=(1<<EN); //

PORT\_CTL &=~(1<<STK); //

for(x = 0;x < 4; x++)

{

SendData(Display[x][y],Display[x][y+8]);

}

LineWrite(y);

PORT\_CTL |=(1<<STK); //

PORT\_CTL &=~(1<<EN); //

\_delay\_us(10); //延时

}

}

void Ledmove()

{

unsigned char i = 0;

unsigned char col = 0;

static unsigned char buf[16] = {0};

for(i=0;i<16;i++)

{

for(col=0;col<18;col++)

{

if(col == 0) //

{

if(Display[col][i] & 0x01)

{

buf[i] |= 0x80;

}

buf[i] >>= 1;

Display[col][i] >>= 1;

}

else

{

if(Display[col][i] & 0x01) //0X01

{

Display[col-1][i] |= 0x80; //0X80

}

Display[col][i] >>= 1;

}

}

if(buf[i] & 0x01)

{

Display[17][i] |= 0X80; //0X80

}

buf[i] >>= 1;

}

}

/////////////////////////////////////////////////////////////////

void Timeconfig()//定时器初始化

{

TCCR0 |= (1<<CS00)|(1<<CS02);//普通计数模式，1/1024 控制时间

TCNT0 = 0X00; //赋值

SREG=0X80; //全局中断打开

TIMSK |= (1<<TOIE0); //定时器0溢出中断使能

sei();

}

ISR(TIMER0\_OVF\_vect)

{ // user code here

TCNT0 = 0xd9; //0.0010s 1/4000000/1024\*(0xff-TCNT0) 39

T0\_Counter++;

TIFR |= \_BV(TOV0);

if(T0\_Counter >= 100)//1S FLAG=1

{

Flag=1;

T0\_Counter=0;

}

}

void renew(unsigned char b)

{

static unsigned char disbuf[16]={0};

unsigned char i,col=0;

if(b == 0)

{

for(i=0;i<16;i++) //时间

{

Display[18][i] = Number[23][i];

Display[17][i] = Number[23][i];

Display[16][i] = Number[23][i];

Display[15][i] = Number[23][i];

Display[14][i] = Number[23][i];

Display[13][i] = Number[22][i];

Display[12][i] = Number[21][i];

Display[11][i] = Number[20][i];

Display[10][i] = Number[19][i];

Display[9][i] = Number[18][i];

Display[8][i] = Number[17][i];

Display[7][i] = Number[s%10][i];

Display[6][i] = Number[s/10][i];

Display[5][i] = Number[10][i];

Display[4][i] = Number[m%10][i];

Display[3][i] = Number[m/10][i];

Display[2][i] = Number[10][i];

Display[1][i] = Number[h%10][i];

Display[0][i] = Number[h/10][i];

}

}

if(b == 7)

{

for(i=0;i<16;i++) //日期

{

Display[16][i] = Number[23][i];

Display[15][i] = Number[23][i];

Display[14][i] = Number[23][i];

Display[13][i] = Number[16][i];

Display[12][i] = Number[15][i];

Display[11][i] = Number[d % 10][i];

Display[10][i] = Number[d / 10][i];

Display[9][i] = Number[14][i];

Display[8][i] = Number[13][i];

Display[7][i] = Number[moth % 10][i];

Display[6][i] = Number[moth / 10][i];

Display[5][i] = Number[12][i];

Display[4][i] = Number[11][i];

Display[3][i] = Number[y % 10][i];

Display[2][i] = Number[y / 10 % 100][i];

Display[1][i] = Number[y / 100 % 10][i];

Display[0][i] = Number[y / 1000][i];

}//Ledmove();

}

if(b == 18)

{

for(i=0;i<16;i++) //制作

{

Display[17][i] = zhizuo[17][i];

Display[16][i] = zhizuo[16][i];

Display[15][i] = zhizuo[15][i];

Display[14][i] = zhizuo[14][i];

Display[13][i] = zhizuo[13][i];

Display[12][i] = zhizuo[12][i];

Display[11][i] = zhizuo[11][i];

Display[10][i] = zhizuo[10][i];

Display[9][i] = zhizuo[9][i];

Display[8][i] = zhizuo[8][i];

Display[7][i] = zhizuo[7][i];

Display[6][i] = zhizuo[6][i];

Display[5][i] = zhizuo[5][i];

Display[4][i] = zhizuo[4][i];

Display[3][i] = zhizuo[3][i];

Display[2][i] = zhizuo[2][i];

Display[1][i] = zhizuo[1][i];

Display[0][i] = zhizuo[0][i];

}

}

}

void main()

{

unsigned char a=0,b=0,c=0,d=0;

int count=0;

LEDconfig();

Timeconfig();

while(1)

{

if(count++ >= 5) //控制移动速度

{

count = 0;

Ledmove();

}

display();

if(Flag == 1) //1s 显示时间

{

Time\_counter();

a+=1;

Flag = 0;

if(a == 5||b == 40) //XX制作

{

c =55;

renew(18);

}

if (c == 10) //1\*10=10s 更新显示日期

{

d = 60;

renew( 7 ); //送日期数据

}

if(d == 20 )

{

a=20;

b=75;

renew(0); //时间数据

}

b--;

c--;

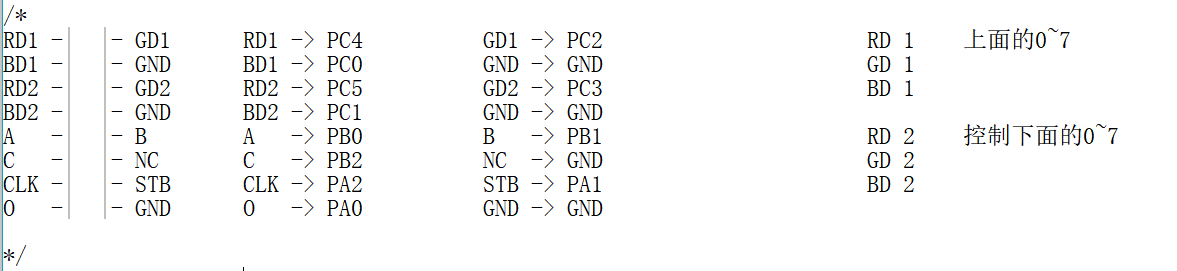
d--;

}

}

}

#main.h



#define EN PA0

#define STK PA1

#define CLK PA2

#define RowA PB0

#define RowB PB1

#define RowC PB2

#define B1 PC0

#define B2 PC1

#define G1 PC2//PC5//PA3

#define G2 PC3//PC3//PA5

#define R1 PC4//PC4//PA6

#define R2 PC5

#define DDR\_LINE DDRB // 扫描行端口定义

#define PORT\_LINE PORTB

#define DDR\_DATA1 DDRC //DDRC // 数据端口定义

#define DDR\_DATA2 DDRC //DDRC // 数据端口定义

#define PORT\_DATA1 PORTC//PORTC

#define PORT\_DATA2 PORTC

#define DDR\_CTL DDRA

#define PORT\_CTL PORTA

// LED显示屏说明

// 屏I数据由R1/G1提供

// 屏II数据由R2/G2提供

// LA LB LC LD决定显示屏I/屏II的第几行

// 每块屏有16行

#define CLK\_1 PORTA |= (1<<PA2)

#define CLK\_0 PORTA &=~(1<<PA2)

#font\_library.h

unsigned Display[18][16]=//显示缓冲区

{

};

unsigned char Number[30][16]={

// 0(0) 1(1) 2(2) 3(3) 4(4) 5(5) 6(6) 7(7) 8(8) 9(9)

// 取模方式 阴码 逆向 逐行式

//显示时间 字库

{0x00,0x00,0x00,0x18,0x24,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x24,0x18,0x00,0x00},/\*"0",0\*/

{0x00,0x00,0x00,0x10,0x1C,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x7C,0x00,0x00},/\*"1",1\*/

{0x00,0x00,0x00,0x3C,0x42,0x42,0x42,0x40,0x20,0x10,0x08,0x04,0x42,0x7E,0x00,0x00},/\*"2",2\*/

{0x00,0x00,0x00,0x3C,0x42,0x42,0x40,0x20,0x18,0x20,0x40,0x42,0x42,0x3C,0x00,0x00},/\*"3",3\*/

{0x00,0x00,0x00,0x20,0x30,0x30,0x28,0x24,0x24,0x22,0xFE,0x20,0x20,0xF8,0x00,0x00},/\*"4",4\*/

{0x00,0x00,0x00,0x7E,0x02,0x02,0x02,0x1E,0x22,0x40,0x40,0x42,0x22,0x1C,0x00,0x00},/\*"5",5\*/

{0x00,0x00,0x00,0x18,0x24,0x02,0x02,0x3A,0x46,0x42,0x42,0x42,0x44,0x38,0x00,0x00},/\*"6",6\*/

{0x00,0x00,0x00,0x7E,0x42,0x20,0x20,0x10,0x10,0x08,0x08,0x08,0x08,0x08,0x00,0x00},/\*"7",7\*/

{0x00,0x00,0x00,0x3C,0x42,0x42,0x42,0x24,0x18,0x24,0x42,0x42,0x42,0x3C,0x00,0x00},/\*"8",8\*/

{0x00,0x00,0x00,0x1C,0x22,0x42,0x42,0x42,0x62,0x5C,0x40,0x40,0x24,0x18,0x00,0x00},/\*"9",9\*/

{0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x18,0x00,0x00,0x00,0x00,0x18,0x18,0x00,0x00},/\*":",10\*/

{0x08,0x08,0xF8,0x08,0x04,0x02,0xF8,0x08,0x08,0x08,0xFF,0x00,0x00,0x00,0x00,0x00},/\*"年",11\*/

{0x00,0x00,0x3F,0x01,0x01,0x01,0x1F,0x01,0x01,0x01,0x7F,0x01,0x01,0x01,0x01,0x00},/\*"年",12\*/

{0x00,0xF8,0x08,0x08,0x08,0xF8,0x08,0x08,0x08,0xF8,0x08,0x08,0x08,0x04,0x02,0x01},/\*"月",13\*/

{0x00,0x1F,0x10,0x10,0x10,0x1F,0x10,0x10,0x10,0x1F,0x10,0x10,0x10,0x10,0x14,0x08},/\*"月",14\*/

{0x00,0xF8,0x08,0x08,0x08,0x08,0x08,0xF8,0x08,0x08,0x08,0x08,0x08,0x08,0xF8,0x00},/\*"日",15\*/

{0x00,0x0F,0x08,0x08,0x08,0x08,0x08,0x0F,0x08,0x08,0x08,0x08,0x08,0x08,0x0F,0x00},/\*"日",16\*/

{0x00,0xF8,0x08,0xF8,0x08,0xF8,0x80,0x88,0xF8,0x84,0x82,0xF8,0x80,0x80,0xFE,0x00},/\*"星",17\*/

{0x00,0x0F,0x08,0x0F,0x08,0x0F,0x00,0x00,0x1F,0x00,0x00,0x0F,0x00,0x00,0x3F,0x00},/\*"星",18\*/

{0x44,0x44,0xFE,0x44,0x44,0x7C,0x44,0x44,0x7C,0x44,0x44,0xFF,0x20,0x44,0x82,0x41},/\*"期",19\*/

{0x00,0x3E,0x22,0x22,0x22,0x3E,0x22,0x22,0x22,0x3E,0x22,0x22,0x21,0x21,0x28,0x10},/\*"期",20\*/

{0x00,0xFE,0x40,0x40,0x40,0x40,0xFC,0x20,0x20,0x20,0x20,0x10,0x10,0x10,0xFF,0x00},/\*"五",21\*/

{0x00,0x3F,0x00,0x00,0x00,0x00,0x0F,0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x7F,0x00},/\*"五",22\*/

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/\*" ",23\*/

};

unsigned zhizuo[20][16] =

{

{0x00,0x00,0x00,0x10,0x1C,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x7C,0x00,0x00},/\*"1",0\*/

{0x00,0x00,0x00,0x10,0x1C,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x7C,0x00,0x00},/\*"1",1\*/

{0x00,0x00,0x00,0x20,0x30,0x30,0x28,0x24,0x24,0x22,0xFE,0x20,0x20,0xF8,0x00,0x00},/\*"4",2\*/

{0x00,0x00,0x00,0x20,0x30,0x30,0x28,0x24,0x24,0x22,0xFE,0x20,0x20,0xF8,0x00,0x00},/\*"4",3\*/

{0x00,0x00,0x00,0x10,0x1C,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x7C,0x00,0x00},/\*"1",4\*/

{0x00,0x00,0x00,0x3F,0x42,0x42,0x42,0x42,0x3E,0x02,0x02,0x02,0x02,0x07,0x00,0x00},/\*"P",5\*/

{0x00,0x00,0x00,0x18,0x24,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x24,0x18,0x00,0x00},/\*"0",6\*/

{0x00,0x00,0x00,0x20,0x30,0x30,0x28,0x24,0x24,0x22,0xFE,0x20,0x20,0xF8,0x00,0x00},/\*"4",7\*/

{0x20,0xFC,0x20,0x20,0xFF,0x80,0xF8,0x88,0x88,0xF8,0x88,0x88,0xF8,0x20,0x10,0x08},/\*"黄",8\*/

{0x02,0x1F,0x02,0x02,0x7F,0x00,0x0F,0x08,0x08,0x0F,0x08,0x08,0x0F,0x02,0x04,0x08},/\*"黄",9\*/

{0xF8,0x08,0xF8,0x08,0xF8,0x80,0xFF,0x00,0xF8,0x08,0x08,0xF8,0x80,0x84,0xA2,0x40},/\*"景",10\*/

{0x0F,0x08,0x0F,0x08,0x0F,0x00,0x7F,0x00,0x0F,0x08,0x08,0x0F,0x00,0x10,0x20,0x00},/\*"景",11\*/

{0x80,0x80,0x9F,0x84,0xE4,0x84,0x84,0xDF,0xC4,0xA4,0xA4,0x9C,0x87,0x82,0x80,0x80},/\*"琳",12\*/

{0x08,0x08,0x08,0x08,0x3D,0x08,0x08,0x1C,0x1D,0x2A,0x2A,0x49,0x08,0x08,0x08,0x08},/\*"琳",13\*/

{0x20,0x24,0x24,0xFC,0x22,0x20,0xFF,0x20,0x20,0xFC,0x24,0x24,0x64,0xA4,0x20,0x20},/\*"制",14\*/

{0x20,0x20,0x20,0x25,0x24,0x24,0x27,0x24,0x24,0x25,0x25,0x25,0x21,0x20,0x28,0x10},/\*"制",15\*/

{0x90,0x90,0x90,0x88,0x48,0x4C,0x2C,0x0A,0x09,0x08,0x08,0x08,0x08,0x08,0x08,0x08},/\*"作",16\*/

{0x00,0x00,0x00,0x7F,0x01,0x01,0x01,0x1F,0x01,0x01,0x01,0x3F,0x01,0x01,0x01,0x01},/\*"作",17\*/

};