西安电子科技大学

《单片机应用系统创作设计》 课程实验报告

实验名称:多功能电子时钟设计

电子工程 学院 1602012 班

姓名 杨文傲 学号 16020120020

同作者 任浩坤（16020120017）

同作者 任子晗（16020120018）

同作者 陈凯伟（16020120019）

同作者

成 绩

实验报告日期 2019 年 6 月 1 日

指导教师评语:

指导教师:

年 月 日

实验报告内容基本要求及参考格式

一、实验目的

二、实验所用仪器（或实验环境）

三、实验基本原理及步骤（或方案设计及理论计算）

四、实验数据记录（或仿真及软件设计）

五、实验结果分析及回答问题（或测试环境及测试结果）

摘 要

时钟是生活中常用的一种计时器，电子钟亦称数显钟，是一种用数字电路技术实现时、分、秒计时的装置，与机械时钟相比，直观性为其主要显著特点，且因非机械驱动，具有更长的使用寿命，相较石英钟的石英机芯驱动，更具准确性。

我们设计的多功能电子时钟不仅具有查看时间的功能，还用了尽可能少的元器件完成了闹钟、计时、重置等多种功能。

关键词： 电子时钟　多功能　MSP430　简便

第一章 绪论

1.1 选题背景及其意义

随着社会经济活动的日益频繁和现代生活节奏的不断加快，人们的时间观念越来越强，虽然新时代科技产品如手机、计算机等已经具备了时钟的功能，但是传统时钟却始终独具一格，没有被淘汰，各种以时钟为基础的产品却源源不断的在推陈出新，可谓一大热门行业。相比于机械时钟更注重外观和价格，朴素的电子时钟能为跟多人所用，在人们的日常生活中发挥着巨大的作用，所以，人们也追求折更简便、功能更多的电子时钟。

1.2 研究目的及其意义

我们所研究的产品主要的功能是计时与显示时间，目的是为了让我们更有效率的规划自己的日常生活，达到事半功倍的效果。

具有以下功能：

显示系统时间（年、月、日、分、秒）；

设置系统时间；

以闹钟方式提醒设定时间；

以计时方式提醒设定时间。

1.3 研究任务

本次实验实现利用单片机 MSP430G2553。根据设计要求进行程序编写，并烧录入单片机。通过键盘控制输出参数，并在 LED12864 液晶屏上实时显示；并控制蜂鸣器在固定时间响起。

第二章 系统工作任务及工作指标

2.1 工作指标

本文设计的多功能电子时钟的功能有四种：显示系统时间（年、月、日、时、分、秒）；设置系统时间；以闹钟方式提醒设定时间；以计时方式提醒设定时间。

本设备配置了 16 个按键，分别用于设定时间十个数字，工作模式，重置，返回主菜单，返回上一层，删除，确定。

采用 MSP430G2553 单片机为控制器芯片，全数字化控制，人机界面友好；采用液晶显示器（LCD）；菜单式操作，四种功能一目了然；可调整时间流逝的快慢。保证用户及仪器的安全；用户设置一次便勿需费心计时，且完成时有蜂鸣声音提示。

2.2 工作任务

本文设计包括硬件设计和软件设计两部分。硬件部分模块相对简单包括电源设计，烧录部分，键盘设计，显示电路，输出电路。

软件设计相对复杂，开机后系统先进行初始化，显示欢迎语，然后进行设置端口，规定每个端口的输入输出状态，重复键盘按键扫描，在扫描到按键后，返回相应的标志位，对标志位进行判断，不同的标志位进入不同的函数。功能 1 设置时间具体流程为：在通电显示欢迎语后，按下 MODE 键再按 0，进入设定时间流程，按时分秒依次键入设定时间，按 ENTER 确定，以此系统时间开始计时。功能 2 倒计时并蜂鸣器响，具体流程为：由上步，直接按 MODE 再按 1，直接开始倒计时，时间到蜂鸣器响。功能 3 闹钟，具体流程为：按下 MODE 再按数字 2 进入闹钟模式，依次键入时分秒，等待，系统时钟与设定时钟一致时，蜂鸣器响。

第三章 主要电路元器件简介

3.1 MSP430G2553

3.1.1.工作原理

MSP430 系列单片机是美国德州仪器（TI）1996 年开始推向市场的一种 16 位超低功耗、具有精简指令集（RISC）的混合信号处理器（Mixed Signal Processor）。

MSP430 单片机被称为混合信号处理器，是由于其针对实际应用需求，将多个不同功能的模拟电路、数字电路模块和微处理器集成在一个芯片上，以提供“单片机”解决方案。该系列单片机多应用于需要电池供电的便携式仪器仪表中。

MSP430 系列单片机的各系列都集成了较丰富的片内外设,有看门狗（WDT）、模拟比较器 A、定时器 A0、定时器 A1、定时器 B0、UART、SPI、I2C、硬件乘法器、液晶驱动器、10 位/12 位 ADC、16 位 Σ-Δ ADC、DMA、I/O 端口、基本定时器、实时时钟和 USB 控制器等若干外围模块的不同组合。其中，看门狗可以使程序失控时迅速复位；模拟比较器进行模拟电压的比较，配合定时器，可设计出 A/D 转换器；16 位定时器具有捕获/比较功能，大量的捕获/比较寄存器，可用于事件计数、时序发生、PWM 等；有的器件更具有可实现异步、同步及多址访问串行通信接口可方便的实现多机通信等应用；具有较多的 I/O 端口，P0、P1、P2 端口能够接收外部上升沿或下降沿的中断输入；10/12 位硬件 A/D 转换器有较高的转换速率，最高可达 200kbps ，能够满足大多数数据采集应用；能直接驱动液晶多达 160 段；实现两路的 12 位 D/A 转换；硬件 I2C 串行总线接口实现存储器串行扩展；以及为了增加数据传输速度，而采用的 DMA 模块。MSP430 系列单片机的这些片内外设为系统的单片解决方案提供了极大的方便。

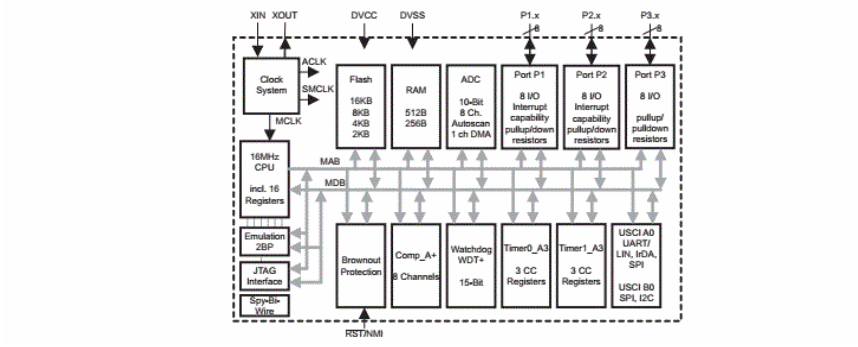


图 3.1 MSP430G2553 功能方框图

3.1.2. 主要特性参数

- 低压电源范围：1.8V 至 3.6V；
- 功耗：

  运行模式:230μA

  待机模式:0.5μA

  关机模式:0.1μA；

器件引出脚配置、MSP430G2x13 和 MSP430G2x53、20 引脚器件、 TSSOP 和 PDIP 封装

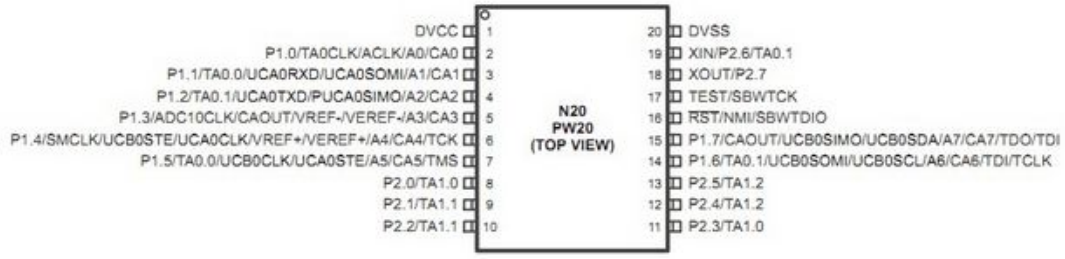


图 3.2 MSP430G2553 引脚图

3.1.3. MSP430G2553 引脚功能简介

- DVCC 正数字电源电压
- DVSS 数字电路地
- TEST 为端口一上的 JTAG 引脚选择测试模式
- P1.4/A4/TCK 时钟
- P1.5/A5/TMS 模式选择
- P1.6/TDI 数据输入
- P1.7/TDO 数据输出
- RST 复位信号输入
- P1.1/A1/RXD 接受数据
- P1.2/A2/TXD 发送数据
- P1.0/A0 通用数字 I/O 引脚
- P1.3/A3 通用数字 I/O 引脚
- P2.0—2.5 标准输入 I/O 或者访问时的高八位地址引脚
- P2.6/XIN 通用数字 I/O 引脚
- P2.7/XOUT 通用数字 I/O 引脚

3.2 OLED

OLED 是一种字符集图形点阵液晶显示器,它主要由行驱动器/ 列驱动器及128×64 全点阵液晶显示器组成。可完成图形显示,也可以显示 7.5×2 个(16×16 点阵)汉字，与外部 CPU 接口采用并行或串行方式控制.

- 串口通信接口管脚信号如表 3.2 所示：

- 表 3.2 串口通信引脚功能

管脚号	名称	LEVER	功能
1	GND	0V	电源地
2	VDD	+5V	电源正(3.0V-5.5V)
3	SCL	-	串行时钟
4	SDA	H/L	串行数据

第四章 硬件设计

硬件原理硬件电路的设计决定一个系统的功能，是设计的基础所在，而一般设计的目标：可靠，简洁，高效，优化，好的硬件电路可以给程序的编写带来极大的优势，同时使可以很好的提高该信号设计的精度和灵敏度，使整个系统工作协调有序。

4.1 硬件原理框图

对于该多功能电子时钟的设计，我们采用了以 MSP430G2553 芯片作为核心处理器，编程实现各种不同类型数据串口调用，最后通过输出端口输出。结构简单，思路井井有条，而根据设计的基本要求，我们又把其细分为不同的功能模块，各个功能模块相互联系，相互协调，通过单片机程序构成一个统一的整体，其整体电路原理框图如图 4.1 所示：

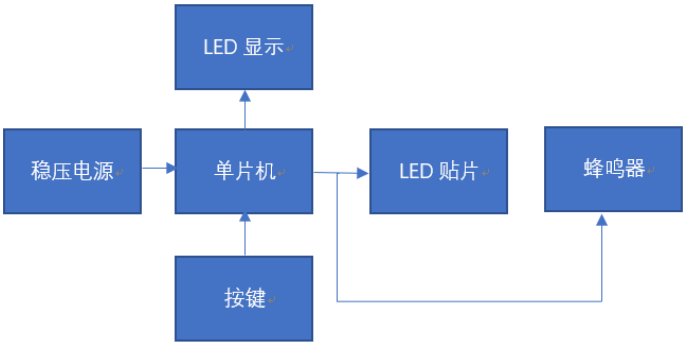


图 4.1 系统方案框图

4.2 单片机系统设计

MSP430G2553 单片机是该信号发生器的核心，具有两个 16 位定时器，5 种低功耗，多通道 10-14 位 AD 转换器，比较器，液晶驱动器，电压检测，FLASH 储存器。由于本设计功能简单，数据处理容易，数据存储空间也足够，因为我们采用了片选法选择芯片，进行芯片的选择和地址的译码。

单片机主要引脚分配如下：

- TMS、TCK：JTAG 测试模式选择、JTAG 测试时钟；
- RST：复位电路；
- TDI、TDO：与显示屏数据传输、JTAG 数据输入输出；

- P2 口：键盘输入信号；

4.3 供电模块电路

在电源部分，本设计需要实现+5V、+3.3V 这几种电压，+5V 电压主要应用于各个元件的 VCC,+3.3V 给 MSP430 进行供电.具体的电路实现方法如下图 4.2 所示：

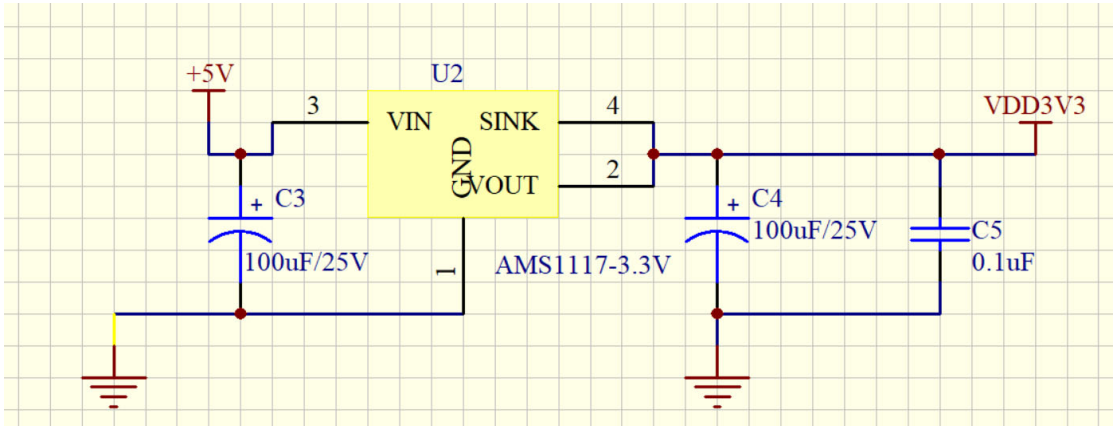


图 4.2 电源模块电路原理图

其中使用 AMS1117-3.3 进行稳压，将 5V 输入电压稳压为 3.3V.

4.4 键盘模块电路

由于需要按键较多，故本设计采用了 4*4 按键组合键盘，电路原理图如图 4.3 所示，其中：

- K1：数字 0
- K2：数字 4
- K3：数字 8
- K4：回到主界面
- K5：数字 1
- K6：数字 5
- K7：数字 9
- K8：SKIP
- K9：数字 2
- K10：数字 6
- K11：模式选择

- K12：返回上一步
- K13：数字 3
- K14：数字 7
- K15：重置
- K16：确定

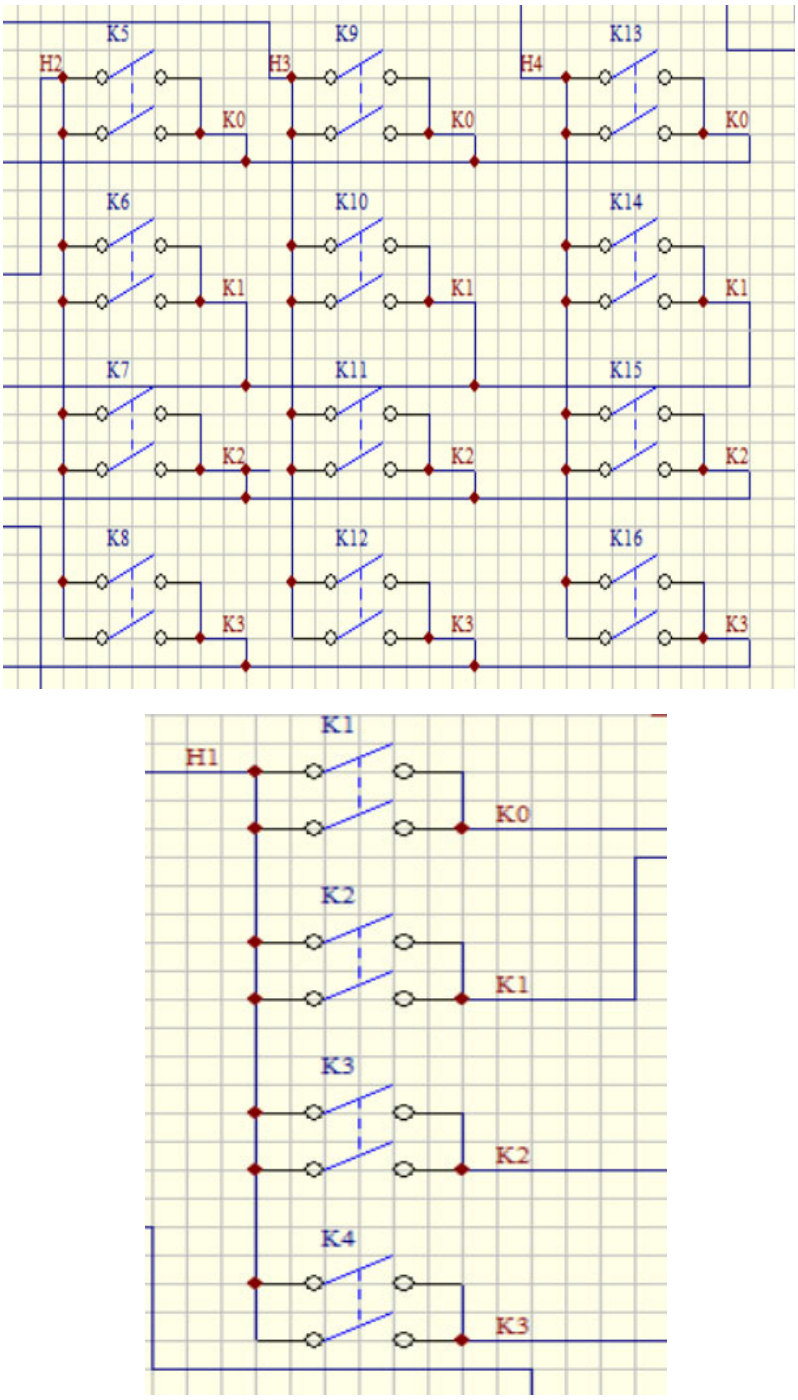


图 4.3 组合键盘模块

4.5 总体设计电路

总体的设计电路如图 4.4 所示：

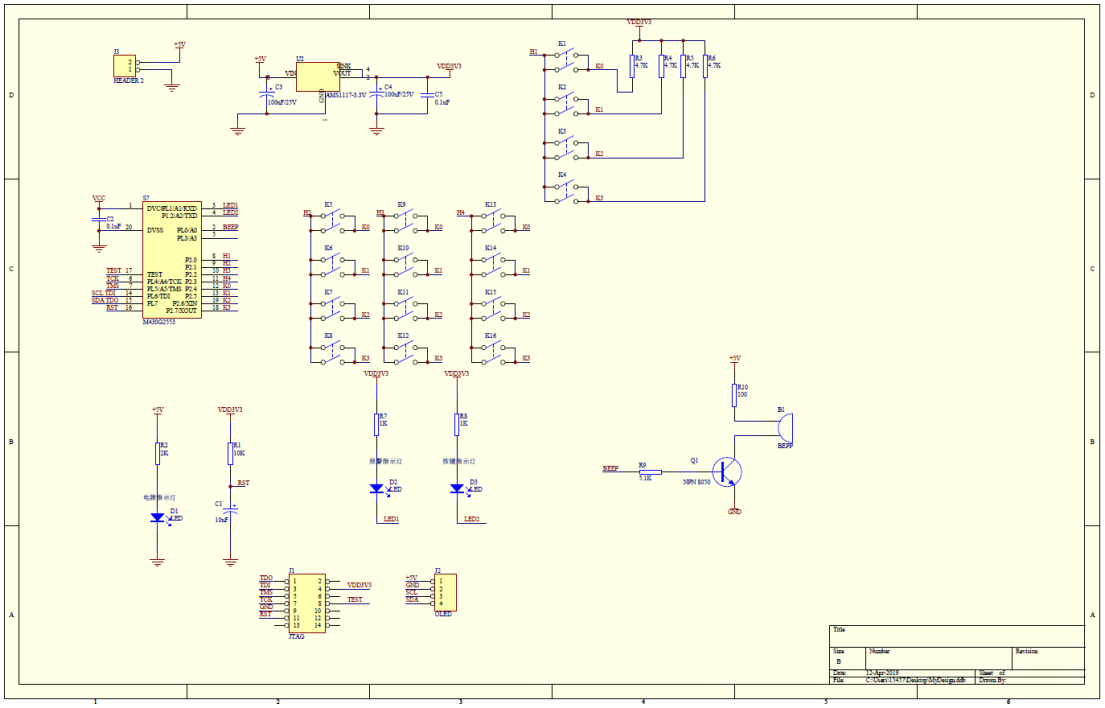


图 4.4　总体电路原理图

第五章 系统软件设计

5.1 主程序设计

开机后系统先进行初始化：首先设置系统时钟，设定 LFXT1 为 vol 时钟为 12KHZ，设置捕获/比较寄存器，初始值为 12000，对于 ACLK 时钟频率为 12khz 的频率，相当于 1s。打开计时中断，进行按键扫描：若为 Mode0 则设定时间；若为 Mode1 则进行倒计时；若为 Mode2 则为设定闹钟。主程序框图如图 5.1 所示：

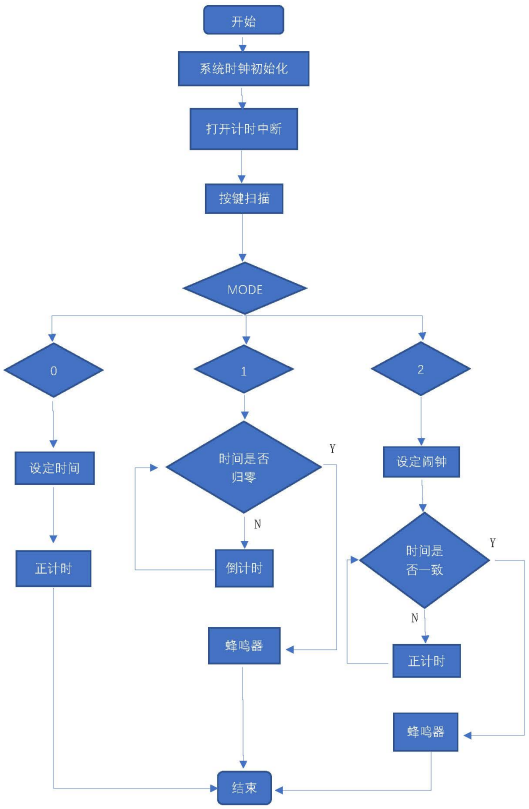


图 5.1 主程序流程图

5.2 设定时间模块设计

将时分秒初始化，扫描按键，依次输入 shi1;shi2;fen1;fen2;miao1;miao2,完成

后按下 ENTER，把此时间设定为系统时间。

5.3 正计时和倒计时模块设计

当频率设定为 13200HZ 时，系统内部定时器计时为 1s,将其发出的信号设为中断，保证每 1s 都引用一次计算正确时间的函数，具体功能是秒每 60 进一，分每 60 进一，时每 24 清零。倒计时功能同理，让参数自加一改为自减一即可，最后将其显示在液晶屏对应位置上。

5.4 闹钟模块的设计

先引用一遍 Mode0 设计时间功能,把设置的时分秒转化为秒数存在变量 shijian1 中，同时在终端中加入计算 shijian2 的函数当两变量相等时，显示时间已到提示字符，同时蜂鸣器响起。

5.5 键盘扫描程序

设定 P2 端口低四位为输出，初始状态为 0000，高四位为输入，循环扫描低四位端口，使被扫描的那位输出 1，当按键按下时，相对应的输入端口会变成 1，再检测输入端口哪一位为 1，即可确定按键位置，以此返回相应 key_code,达成目的。在此基础上还有防抖动功能，通过延迟 200ms 再次检测是否有按键按下，防止误触。

5.6 其余按键功能设计

ESC:清空屏幕设定时间的一行，清空所有标志位，返回程序最开始。

RESET：清空屏幕设定时间的一行，清空系统时间，清空所有标志位，返回程序最开始。

BACK：当处于 Mode 时，返回主菜单，当处于 Mode0,1,2 时返回 Mode,具体做法是清空并修改相应标志位。

ENTER:把设定时间置为系统时间。

结束语

基于 MSP30G2553 的电子时钟的设计，使我们整个小组的成员初步了解并掌握了单片机的设计和实现过程，对其软硬结合的方式有了更深刻的体会，对于我们日后将所学各门课程知识结合起来有巨大的帮助。同时，这次设计耗时较大，锻炼了我们分工合作，解决一个相对具体问题的能力，对我们日后步入社会，走上工作岗位也有巨大的帮助，总而言之，这是一次非常有意义的选修课。

致谢

本次设计论文是在导师楼顺天教授的悉心指导下完成的，其严谨的治学态度和科学的工作方法给了我们极大的帮助和影响。在此由衷感谢楼顺天老师对我们的关心和指导。

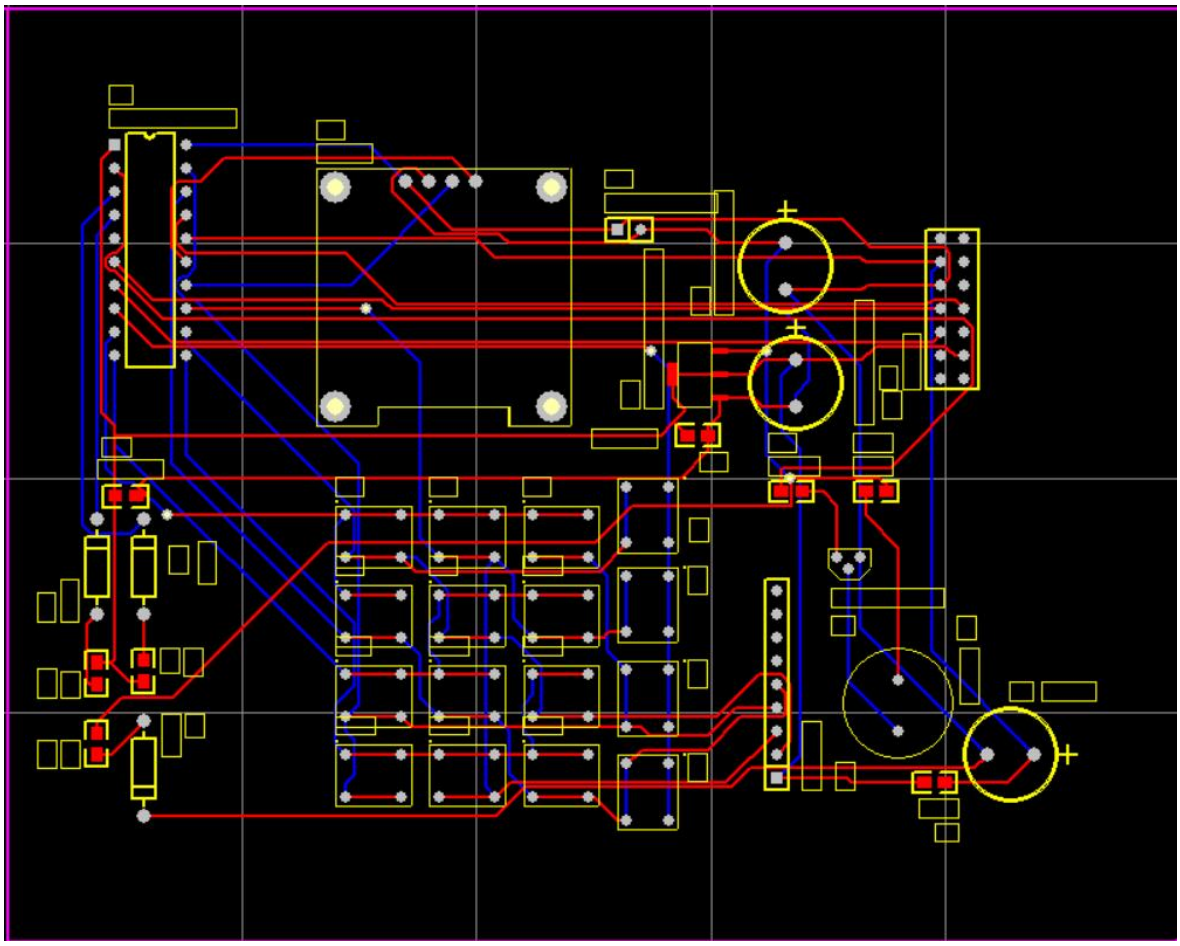
参考文献

[1] 张晞，王德银，张晨，MSP430 系列单片机实用 C 语言程序设计[M]，北京：人民邮电出版社，2005.9

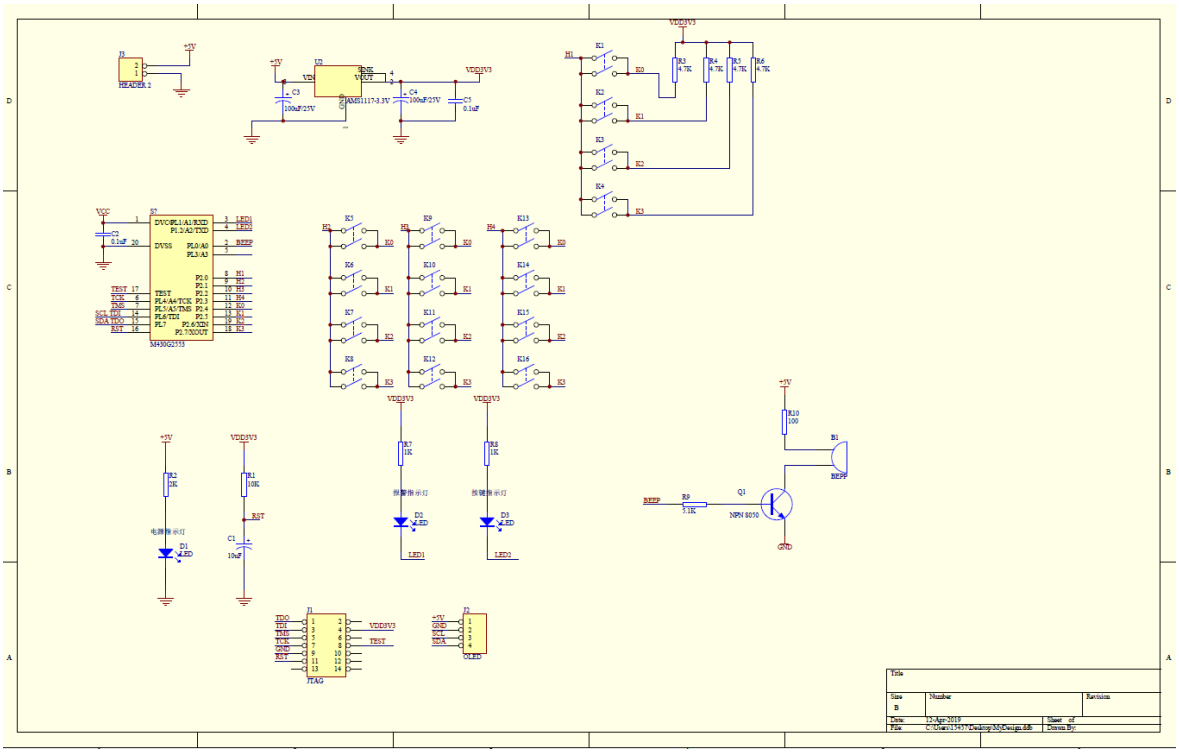
[2] MSP430G2553 数据手册[S]，德州仪器 半导体技术（上海）有限公司

[3] 孙肖子，模拟电子电路技术基础（第二版）[M].西安电子科技大学出版社，2008

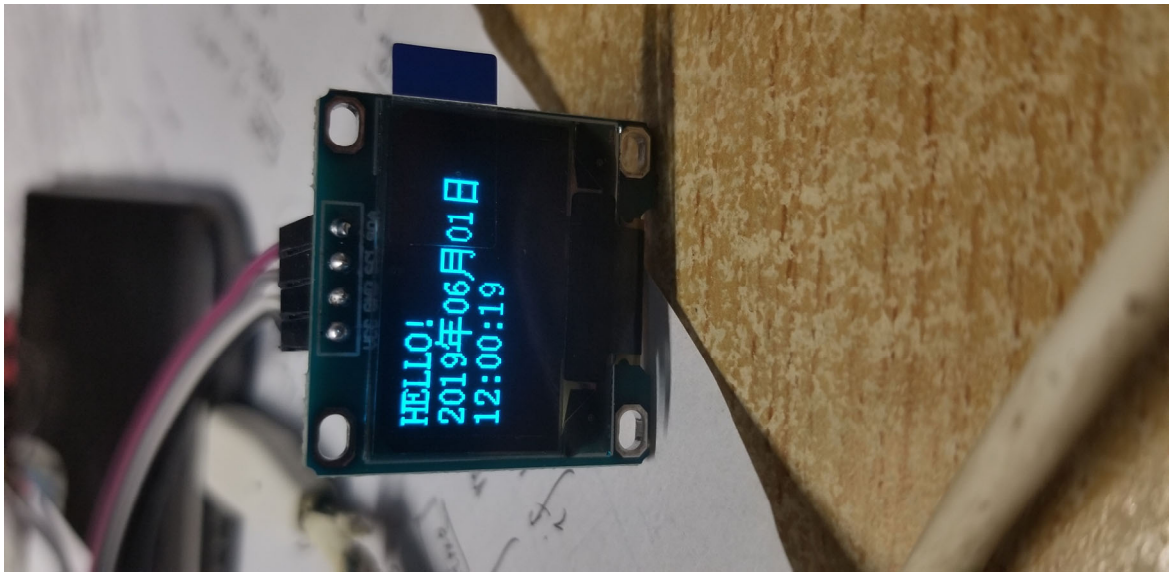
附录



附图 1.1 PCB

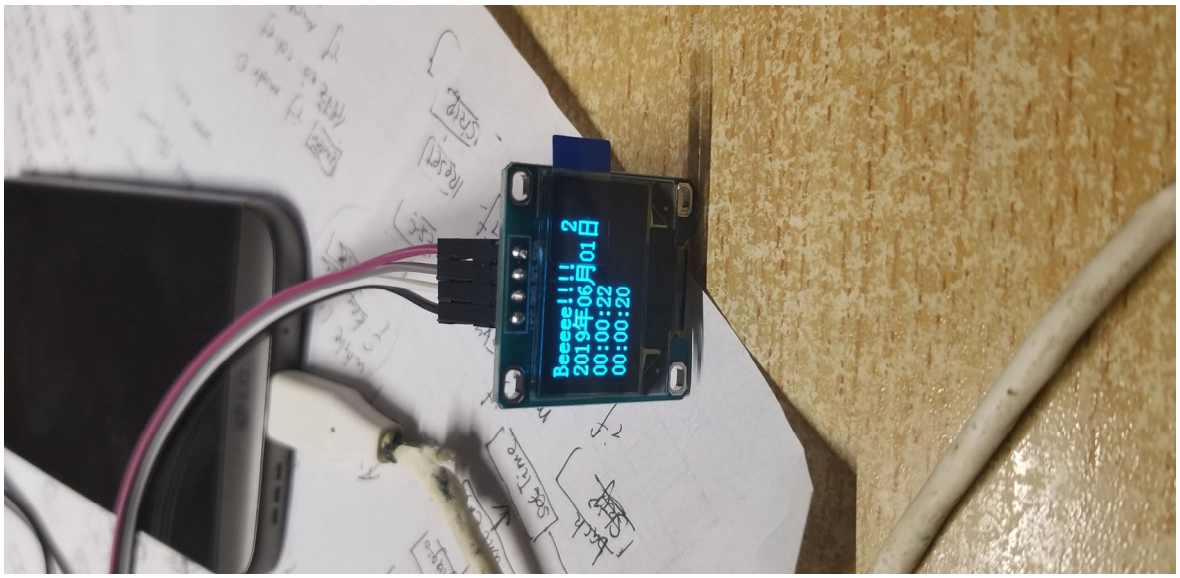


附图 1.2 电路原理图



附图 1.3 设置时间

附图 1.4 倒计时



附图 1.5 闹钟

附录 2

主函数代码

```
//#include "io430.h"
//#include "oled.h"
#include <msp430.h>
#include "OLED.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "key2.h"

#define uchar      unsigned char
#define uint  unsigned int
#define ulong      unsigned long

#define CPU_F ( (double) 8000000)
#define delayus( x )   __delay_cycles( (long) (CPU_F * (double) x / 1000000.0) )
#define delayms( x )  __delay_cycles( (long) (CPU_F * (double) x / 1000.0) )

//unsigned char Key_scan (void);
void date(void);
void Time(void);
void revTime(void);
void showTime(void);


unsigned char temp;

unsigned char nian1,nian2,nian3,nian4,yue1,yue2,ri1,ri2;
unsigned char miao1=0x30;
unsigned char fen1=0x30;
unsigned char shi1=0x30;
unsigned char miao2=0x30;
unsigned char fen2=0x31;
unsigned char shi2=0x30;              //起始时间 00:00:00
int time_flag=1;
int key_flag=1;

/*
```

```
unsigned char Key_scan (void)
{
char scan_sig[4] = {0x1F,0x2F,0x4F,0x8F};// 还要改
char msb_sig[4] = {0x10,0x20,0x40,0x80};
char p2_sig[4] = {0x01,0x02,0x04,0x08};
char MSB, LSB, key_code,p2;
//主函数定义 IN OUT 状态设置
// P1 口低 4 位作为行线输出低电平//P2.4 到 P2.7 设为输出并清零
P2OUT = 0xF0;
 temp = (P2IN) & 0x0F; // P2 口低 4 位作为列线输入//P2.0 到 P2.3 设为输入，其他不用变
 if(temp)
 {
delayms(10);//延时，再次读取输入，消除抖动
temp = (P2IN) & 0x0F;
if(temp)// 确实有键按下
{ P2OUT = 0x00;
 //按行扫描
 for (char i = 0;i < 4; i++)
 {
P2OUT = scan_sig[i];
for(char j=0;j<4;j++)
{
   p2=P2IN;
   p2 &= p2_sig[j];
   if (p2)
   {
   MSB = msb_sig[j];
   }
   }
// 只设置 P2 高四位 分别为 0001,0010,0100,1000
// 使 MSB=00010000，以此类推
delayms(5);
temp = P2IN;
temp=temp&0x0F;//P2 低四位给入 temp，temp 高四位为 0
if (temp)
{
 //LSB = (~scan_sig[i]) & 0x0F;
 key_code = MSB | temp;
 break;
}
 }
}
else
```

```
 key_code = 0x00; //  无键按下
 }
 else
key_code = 0x00; //  无键按下
 return(key_code);
}
*/

void date(void)              //显示年月日
{
      nian1=0x32;nian2=0x30;nian3=0x31;nian4=0x39;
      yue1=0x30;yue2=0x36;
      ri1=0x30;ri2=0x31;
      OLED_ShowChar( 0, 2, nian1 , 16 );            //2
      OLED_ShowChar( 8, 2, nian2 , 16 );            //0
      OLED_ShowChar( 16, 2, nian3 , 16 );            // 1
      OLED_ShowChar( 24, 2, nian4 , 16 );            //9
      OLED_ShowCHinese(32,2,0);                              //年
      OLED_ShowChar( 48, 2, yue1 , 16 );            //0
      OLED_ShowChar( 56, 2, yue2 , 16 );            // 6
      OLED_ShowCHinese(64,2,1);                              //月
      OLED_ShowChar( 80, 2, ri1 , 16 );       //0
      OLED_ShowChar( 88, 2, ri2 , 16 );       //1
      OLED_ShowCHinese(96,2,2);                              //日
}

void Time(void)              //正数计时
{
      showTime();
      miao2=miao2+1;
      delayms(1000);

      if (miao2>0x39)
      {
            miao2 = 0x30;
            miao1++;
            if (miao1>0x35)
            {
                  miao1 = 0x30;
                  fen2++;
                  if (fen2>0x39)
                  {
                        fen2 = 0x30;
                        fen1++;
```

```
                                        if (fen1>0x35)
                                        {
                                                fen1 = 0x30;
                                                shi2++;
                                                if (shi2>0x39)
                                                {
                                                        shi2 = 0x30;
                                                        shi1++;
                                                        }
                                                }
                                        }
                                }
                        }

        if (shi1 == 0x32 && shi2 == 0x34)
        {
                shi1 = 0x30;
                shi2 = 0x30;
        }               //超过 23：59：59 时清零

}

void revTime(void)              //倒数计时
{
        showTime();
        miao2=miao2-1;
        delayms(200);

        if (miao2<0x30)
        {
                miao2 = 0x39;
                miao1--;
                if (miao1<0x30)
                {
                        miao1 = 0x35;
                        fen2--;
                        if (fen2<0x30)
                        {
                                fen2 = 0x39;
                                fen1--;
                                if (fen1<0x30)
                                {
                                        fen1 = 0x35;
                                        shi2--;
```

```
                                        if (shi2<0x30)
                                        {
                                                shi2 = 0x39;
                                                shi1--;
                                                }
                                        }
                                }
                        }
                }

        if (shi1 == 0x30 && shi2 == 0x30 && fen1 == 0x30 && fen2 == 0x30 && miao1
== 0x30 && miao2 == 0x30)
        {
            time_flag=2;
        }                       //倒计时时间为 0 时将 time_flag 变为 2

}

void showTime(void)
{
        OLED_ShowChar( 0, 4, shi1 , 16 );               //时
        OLED_ShowChar( 8, 4, shi2 , 16 );               //时
        OLED_ShowChar( 16, 4, 0x3A , 16 );                // ：
        OLED_ShowChar( 24, 4, fen1 , 16 );              //分
        OLED_ShowChar( 32, 4, fen2 , 16 );              //分
        OLED_ShowChar( 40, 4, 0x3A , 16 );              // ：
        OLED_ShowChar( 48, 4, miao1 , 16 );             //秒
        OLED_ShowChar( 56, 4, miao2 , 16 );             //秒

}




int main( void )
{
   // Stop watchdog timer to prevent time out reset
        WDTCTL = WDTPW + WDTHOLD;

        if ( CALBC1_8MHZ == 0xFF )              /* If calibration constant erased */
           {
                        while ( 1 )
                                        ;                       /* do not load, trap CPU!! */
```

```
    }
  DCOCTL    = 0;                          /* Select lowest DCOx and MODx settings
*/
  BCSCTL1 = CALBC1_8MHZ;                  /* Set range */
  DCOCTL    = CALDCO_8MHZ;                  /* Set DCO step + modulation * / */

  OLED_Init();                          /* OLED3?ê??¯ */
  OLED_ShowString( 0, 0, "Welcome!" );
  date();

  P2SEL = 0x00;
  P2SEL2= 0x00;
  P1SEL = 0x00;
  P1SEL2= 0x00;

//    P2DIR = 0xF0;
    // P2DIR &= ~BIT1;
    // P2DIR &= ~BIT4;
//    P2REN|=0xFF;
  P1REN|=BIT0;

  P1DIR |= BIT0;
  P1OUT &= ~BIT0;
//    char key;

//    P2DIR=0x0F;
  //  P2REN=0xFF;

  while(1)
  {while(key_flag==1)
  {
  Key_scan();
  }

//    key=b;
  OLED_ShowChar( 0, 6, key_num , 16 );
  key_flag=1;
  }
  while(time_flag==1)
    {
        revTime();

    }
```

```
    return 0;
}
```

Oled.h

```
#include "msp430g2553.h"

#include "oledfont.h"
// ---------------------------------------------------------
// IO 口模拟 I2C 通信
// SCL 接 P2^0
// SDA 接 P2^1

#define SCL_1 P1OUT |= BIT6
#define SCL_0 P1OUT &= ~BIT6
#define SDA_1 P1OUT |= BIT7
#define SDA_0 P1OUT &= ~BIT7

#define  Brightness    0xCF
#define X_WIDTH    128
#define Y_WIDTH    64
#define OLED_CMD    0 //写命令
#define OLED_DATA 1    //写数据

void oled_port_init(void)
{
     P1DIR |= BIT6 + BIT7;
//    P2REN |= BIT0 + BIT1;
//    P2OUT |= BIT0 + BIT1;
}
/**********************OLED    驱    动    程    序    用    的    延    时    程    序
************************************/
void delay(unsigned int z)
{
   unsigned int x,y;
   for(x=z;x>0;x--)
      for(y=100;y>0;y--);
}
/*********************************************
//IIC Start
*********************************************/
void IIC_Start()
{
    SCL_1;
    SDA_1;
    SDA_0;
```

```
    SCL_0;
}

/**********************************************
//IIC Stop
**********************************************/
void IIC_Stop()
{
    SCL_0;
    SDA_0;
    SCL_0;
    SDA_0;
}

/**********************************************
// 通过 I2C 总线写一个字节
**********************************************/
void Write_IIC_Byte(unsigned char IIC_Byte)
{
     unsigned char i;
     for(i=0;i<8;i++)
     {
          if(IIC_Byte & 0x80)
               SDA_1;
          else
               SDA_0;

          SCL_1;
          SCL_0;
          IIC_Byte<<=1;
     }
     SDA_1;
     SCL_1;
     SCL_0;
}


/**********************OLED 写数据************************************/
void OLED_WrDat(unsigned char IIC_Data)
{
     IIC_Start();
     Write_IIC_Byte(0x78);
     Write_IIC_Byte(0x40);              //write data
     Write_IIC_Byte(IIC_Data);
```

```
    IIC_Stop();
}
/**********OLED 写命令**********/
void OLED_WrCmd(unsigned char IIC_Command)
{
    IIC_Start();
    Write_IIC_Byte(0x78);            //Slave address,SA0=0
    Write_IIC_Byte(0x00);            //write command
    Write_IIC_Byte(IIC_Command);
    IIC_Stop();
}
/**********OLED 设置坐标**********/
void OLED_Set_Pos(unsigned char x, unsigned char y)
{
    OLED_WrCmd(0xb0+y);
    OLED_WrCmd(((x&0xf0)>>4)|0x10);
    OLED_WrCmd((x&0x0f)|0x01);
}
/**********OLED 全屏**********/
void OLED_Fill(unsigned char bmp_dat)
{
    unsigned char y,x;
    for(y=0;y<8;y++)
    {
        OLED_WrCmd(0xb0+y);
        OLED_WrCmd(0x01);
        OLED_WrCmd(0x10);
        for(x=0;x<X_WIDTH;x++)
        OLED_WrDat(bmp_dat);
    }
}
/**********OLED 复位**********/
void OLED_CLS(void)
{
    unsigned char y,x;
    for(y=0;y<8;y++)
    {
        OLED_WrCmd(0xb0+y);
        OLED_WrCmd(0x01);
        OLED_WrCmd(0x10);
        for(x=0;x<X_WIDTH;x++)
        OLED_WrDat(0);
    }
}
```

```
void OLED_WR_Byte(unsigned dat,unsigned cmd)
{
     if(cmd)
                {

    OLED_WrDat(dat);

          }
     else {
    OLED_WrCmd(dat);

     }


}
void fill_picture(unsigned char fill_Data)
{
     unsigned char m,n;
     for(m=0;m<8;m++)
     {
          OLED_WR_Byte(0xb0+m,0);      //page0-page1
          OLED_WR_Byte(0x00,0);        //low column start address
          OLED_WR_Byte(0x10,0);        //high column start address
          for(n=0;n<128;n++)
               {
                    OLED_WR_Byte(fill_Data,1);
               }
     }
}

void OLED_DrawBMP(unsigned char x0, unsigned char y0,unsigned char x1, unsigned
char y1,const unsigned char BMP[])
{
 unsigned int j=0;
 unsigned char x,y;

  if(y1%8==0) y=y1/8;
  else y=y1/8+1;
    for(y=y0;y<y1;y++)
    {
         OLED_Set_Pos(x0,y);
     for(x=x0;x<x1;x++)
          {
              OLED_WR_Byte(BMP[j++],OLED_DATA);
```

```
            }
        }
}


void OLED_ShowCHinese(unsigned char x,unsigned char y,unsigned char no)
{
    unsigned char t,adder=0;
    OLED_Set_Pos(x,y);
     for(t=0;t<16;t++)
         {
                OLED_WR_Byte(Hzk[2*no][t],OLED_DATA);
                adder+=1;
       }
         OLED_Set_Pos(x,y+1);
     for(t=0;t<16;t++)
             {
                OLED_WR_Byte(Hzk[2*no+1][t],OLED_DATA);
                adder+=1;
        }
}

//在指定位置显示一个字符,包括部分字符
//x:0~127
//y:0~63
//mode:0,反白显示;1,正常显示
//size:选择字体  16/12
void OLED_ShowChar(unsigned char x,unsigned char y,unsigned char chr,unsigned char
Char_Size)
{
    unsigned char c=0,i=0;
        c=chr-' ';//得到偏移后的值
        if(x>128-1){x=0;y=y+2;}
        if(Char_Size ==16)
            {
            OLED_Set_Pos(x,y);
            for(i=0;i<8;i++)
            OLED_WR_Byte(F8X16[c*16+i],OLED_DATA);
            OLED_Set_Pos(x,y+1);
            for(i=0;i<8;i++)
            OLED_WR_Byte(F8X16[c*16+i+8],OLED_DATA);
            }
            else {
                OLED_Set_Pos(x,y);
```

```
                for(i=0;i<6;i++)
                OLED_WR_Byte(F6x8[c][i],OLED_DATA);

            }
}
//显示一个字符号串
void OLED_ShowString(unsigned char x,unsigned char y,unsigned char *chr)
{
    unsigned char j=0;
    while (chr[j]!='\0')
    {       OLED_ShowChar(x,y,chr[j],16);
            x+=8;
        if(x>120){x=0;y+=12;}
            j++;
    }
}

/**********************OLED 初始化************************************/
void OLED_Init(void)
{
    oled_port_init();
    delay(4000);//初始化之前的延时很重要!
    //delay(500);//初始化之前的延时很重要!
    OLED_WrCmd(0xae);//--turn off oled panel
    OLED_WrCmd(0x00);//---set low column address
    OLED_WrCmd(0x10);//---set high column address
    OLED_WrCmd(0x40);//--set start line address  Set Mapping RAM Display Start Line
(0x00~0x3F)
    OLED_WrCmd(0x81);//--set contrast control register
    OLED_WrCmd(Brightness); // Set SEG Output Current Brightness
    OLED_WrCmd(0xa1);//--Set SEG/Column Mapping     0xa0 左右反置 0xa1 正常
    OLED_WrCmd(0xc8);//Set COM/Row Scan Direction   0xc0 上下反置 0xc8 正常
    OLED_WrCmd(0xa6);//--set normal display
    OLED_WrCmd(0xa8);//--set multiplex ratio(1 to 64)
    OLED_WrCmd(0x3f);//--1/64 duty
    OLED_WrCmd(0xd3);//-set display offset  Shift Mapping RAM Counter (0x00~0x3F)
    OLED_WrCmd(0x00);//-not offset
    OLED_WrCmd(0xd5);//--set display clock divide ratio/oscillator frequency
    OLED_WrCmd(0x80);//--set divide ratio, Set Clock as 100 Frames/Sec
    OLED_WrCmd(0xd9);//--set pre-charge period
    OLED_WrCmd(0xf1);//Set Pre-Charge as 15 Clocks & Discharge as 1 Clock
    OLED_WrCmd(0xda);//--set com pins hardware configuration
    OLED_WrCmd(0x12);
    OLED_WrCmd(0xdb);//--set vcomh
```

```
    OLED_WrCmd(0x40);//Set VCOM Deselect Level
    OLED_WrCmd(0x20);//-Set Page Addressing Mode (0x00/0x01/0x02)
    OLED_WrCmd(0x02);//
    OLED_WrCmd(0x8d);//--set Charge Pump enable/disable
    OLED_WrCmd(0x14);//--set(0x10) disable
    OLED_WrCmd(0xa4);// Disable Entire Display On (0xa4/0xa5)
    OLED_WrCmd(0xa6);// Disable Inverse Display On (0xa6/a7)
    OLED_WrCmd(0xaf);//--turn on oled panel
    OLED_Fill(0x00); //初始清屏
    OLED_Set_Pos(0,0);
}
```

KEY2.h（键盘扫描）

```
//P2DIR = 0x0F;

//uchar key_val;
#include <stdio.h>
#include <stdlib.h>
#define toascii(c) (((unsigned int)(c))&0x7f)
//void delay_nms(unsigned int n);

#define CPU_F ( (double) 8000000)
#define delayus( x )   __delay_cycles( (long) (CPU_F * (double) x / 1000000.0) )
#define delayms( x )  __delay_cycles( (long) (CPU_F * (double) x / 1000.0) )

unsigned char key_num;
unsigned int a=16;
unsigned char b=0x00;
int key_flag;

void Key_scan (void)
{
    char scan_sig[4] = {0x0E,0x0D,0x0B,0x07};
    char MSB, LSB, key_code;
    /*key_code:
    (0x11-k1,0x21-k2,0x41-k3,0x81-k4)
    (0x12-k5,0x22-k6,0x42-k7,0x82-k8)
    (0x14-k9,0x24-k10,0x44-k11,0x84-k12)
    (0x18-k13,0x28-k14,0x48-k15,0x88-k16)
    */
    unsigned short temp;
    P2DIR=0x0F;
        P2OUT = 0x00;//低四位是输出，即 H 是输出
    temp = (~P2IN) & 0xF0;
        if(temp)
    {
        delayms(30);
        temp = (~P2IN) & 0xF0;
        if(temp)
        {
            //P1OUT &=~ BIT2;
            //P1OUT |= BIT3;
```

```
            for (char i = 0;i < 4; i++)
            {
                P2OUT = scan_sig[i];
                delayms(5);
                temp = (~P2IN) & 0xF0;//高四位即 K 是输入
                MSB = temp;
                //MSB = temp<<4;
                if (MSB)
                {
                    LSB = (~scan_sig[i]) & 0x0F;
                    key_code = MSB | LSB;
                    break;
                }
            }
        }
        else key_code = 0x00; // ????
    }
    else key_code = 0x00; // ????
    while(temp)
    {
        temp = (~P2IN) & 0xF0;
    }
    switch(key_code)
    {
        case 0x11:{key_num = 0;break;}
        case 0x21:{key_num = 4;break;}
        case 0x41:{key_num = 8;break;}
        case 0x81:{key_num = 12;break;}
        case 0x12:{key_num = 1;break;}
        case 0x22:{key_num = 5;break;}
        case 0x42:{key_num = 9;break;}
        case 0x82:{key_num = 13;break;}
        case 0x14:{key_num = 2;break;}
        case 0x24:{key_num = 6;break;}
        case 0x44:{key_num = 10;break;}
        case 0x84:{key_num = 14;break;}
        case 0x18:{key_num = 3;break;}
        case 0x28:{key_num = 7;break;}
        case 0x48:{key_num = 11;break;}
        case 0x88:{key_num = 15;break;}
        default: key_num =16;
         }
  P2DIR=0x0F;
/*
```

```
   while(a==16)
   {
    a=key_num;
    b=toascii(a);
   }
*/
            if (key_num!=0x10)
            {
               key_flag=2;
               key_num=key_num+0x30;
            }
//   P2DIR=0x0F;
  /*
   while(a==16)
   {
    a=key_num;
    b=toascii(a);
   }
   */
}
```

Oledfont.h

```
#ifndef __OLEDFONT_H
#define __OLEDFONT_H
//常用 ASCII 表
//偏移量 32
//ASCII 字符集
//偏移量 32
//大小:12*6
/*********************************6*8 的 点 阵
**********************************/


const unsigned char F6x8[][6] =
{

0x00, 0x00, 0x00, 0x00, 0x00, 0x00,// sp

0x00, 0x00, 0x00, 0x2f, 0x00, 0x00,// !
0x00, 0x00, 0x07, 0x00, 0x07, 0x00,// "
0x00, 0x14, 0x7f, 0x14, 0x7f, 0x14,// #
0x00, 0x24, 0x2a, 0x7f, 0x2a, 0x12,// $
0x00, 0x62, 0x64, 0x08, 0x13, 0x23,// %
0x00, 0x36, 0x49, 0x55, 0x22, 0x50,// &
0x00, 0x00, 0x05, 0x03, 0x00, 0x00,// '
0x00, 0x00, 0x1c, 0x22, 0x41, 0x00,// (
0x00, 0x00, 0x41, 0x22, 0x1c, 0x00,// )
0x00, 0x14, 0x08, 0x3E, 0x08, 0x14,// *
0x00, 0x08, 0x08, 0x3E, 0x08, 0x08,// +
0x00, 0x00, 0x00, 0xA0, 0x60, 0x00,// ,
```

```
0x00, 0x08, 0x08, 0x08, 0x08, 0x08,// -
0x00, 0x00, 0x60, 0x60, 0x00, 0x00,// .
0x00, 0x20, 0x10, 0x08, 0x04, 0x02,// /
0x00, 0x3E, 0x51, 0x49, 0x45, 0x3E,// 0
0x00, 0x00, 0x42, 0x7F, 0x40, 0x00,// 1
0x00, 0x42, 0x61, 0x51, 0x49, 0x46,// 2
0x00, 0x21, 0x41, 0x45, 0x4B, 0x31,// 3
0x00, 0x18, 0x14, 0x12, 0x7F, 0x10,// 4
0x00, 0x27, 0x45, 0x45, 0x45, 0x39,// 5
0x00, 0x3C, 0x4A, 0x49, 0x49, 0x30,// 6
0x00, 0x01, 0x71, 0x09, 0x05, 0x03,// 7
0x00, 0x36, 0x49, 0x49, 0x49, 0x36,// 8
0x00, 0x06, 0x49, 0x49, 0x29, 0x1E,// 9
0x00, 0x00, 0x36, 0x36, 0x00, 0x00,// :
0x00, 0x00, 0x56, 0x36, 0x00, 0x00,// ;
0x00, 0x08, 0x14, 0x22, 0x41, 0x00,// <
0x00, 0x14, 0x14, 0x14, 0x14, 0x14,// =
0x00, 0x00, 0x41, 0x22, 0x14, 0x08,// >
0x00, 0x02, 0x01, 0x51, 0x09, 0x06,// ?
0x00, 0x32, 0x49, 0x59, 0x51, 0x3E,// @
0x00, 0x7C, 0x12, 0x11, 0x12, 0x7C,// A
0x00, 0x7F, 0x49, 0x49, 0x49, 0x36,// B
0x00, 0x3E, 0x41, 0x41, 0x41, 0x22,// C
0x00, 0x7F, 0x41, 0x41, 0x22, 0x1C,// D
0x00, 0x7F, 0x49, 0x49, 0x49, 0x41,// E
0x00, 0x7F, 0x09, 0x09, 0x09, 0x01,// F
0x00, 0x3E, 0x41, 0x49, 0x49, 0x7A,// G
0x00, 0x7F, 0x08, 0x08, 0x08, 0x7F,// H
0x00, 0x00, 0x41, 0x7F, 0x41, 0x00,// I
0x00, 0x20, 0x40, 0x41, 0x3F, 0x01,// J
0x00, 0x7F, 0x08, 0x14, 0x22, 0x41,// K
```

```
0x00, 0x7F, 0x40, 0x40, 0x40, 0x40,// L
0x00, 0x7F, 0x02, 0x0C, 0x02, 0x7F,// M
0x00, 0x7F, 0x04, 0x08, 0x10, 0x7F,// N
0x00, 0x3E, 0x41, 0x41, 0x41, 0x3E,// O
0x00, 0x7F, 0x09, 0x09, 0x09, 0x06,// P
0x00, 0x3E, 0x41, 0x51, 0x21, 0x5E,// Q
0x00, 0x7F, 0x09, 0x19, 0x29, 0x46,// R
0x00, 0x46, 0x49, 0x49, 0x49, 0x31,// S
0x00, 0x01, 0x01, 0x7F, 0x01, 0x01,// T
0x00, 0x3F, 0x40, 0x40, 0x40, 0x3F,// U
0x00, 0x1F, 0x20, 0x40, 0x20, 0x1F,// V
0x00, 0x3F, 0x40, 0x38, 0x40, 0x3F,// W
0x00, 0x63, 0x14, 0x08, 0x14, 0x63,// X
0x00, 0x07, 0x08, 0x70, 0x08, 0x07,// Y
0x00, 0x61, 0x51, 0x49, 0x45, 0x43,// Z
0x00, 0x00, 0x7F, 0x41, 0x41, 0x00,// [
0x00, 0x55, 0x2A, 0x55, 0x2A, 0x55,// 55
0x00, 0x00, 0x41, 0x41, 0x7F, 0x00,// ]
0x00, 0x04, 0x02, 0x01, 0x02, 0x04,// ^
0x00, 0x40, 0x40, 0x40, 0x40, 0x40,// _
0x00, 0x00, 0x01, 0x02, 0x04, 0x00,// '
0x00, 0x20, 0x54, 0x54, 0x54, 0x78,// a
0x00, 0x7F, 0x48, 0x44, 0x44, 0x38,// b
0x00, 0x38, 0x44, 0x44, 0x44, 0x20,// c
0x00, 0x38, 0x44, 0x44, 0x48, 0x7F,// d
0x00, 0x38, 0x54, 0x54, 0x54, 0x18,// e
0x00, 0x08, 0x7E, 0x09, 0x01, 0x02,// f
0x00, 0x18, 0xA4, 0xA4, 0xA4, 0x7C,// g
0x00, 0x7F, 0x08, 0x04, 0x04, 0x78,// h
0x00, 0x00, 0x44, 0x7D, 0x40, 0x00,// i
0x00, 0x40, 0x80, 0x84, 0x7D, 0x00,// j
```

```
0x00, 0x7F, 0x10, 0x28, 0x44, 0x00,// k
0x00, 0x00, 0x41, 0x7F, 0x40, 0x00,// l
0x00, 0x7C, 0x04, 0x18, 0x04, 0x78,// m
0x00, 0x7C, 0x08, 0x04, 0x04, 0x78,// n
0x00, 0x38, 0x44, 0x44, 0x44, 0x38,// o
0x00, 0xFC, 0x24, 0x24, 0x24, 0x18,// p
0x00, 0x18, 0x24, 0x24, 0x18, 0xFC,// q
0x00, 0x7C, 0x08, 0x04, 0x04, 0x08,// r
0x00, 0x48, 0x54, 0x54, 0x54, 0x20,// s
0x00, 0x04, 0x3F, 0x44, 0x40, 0x20,// t
0x00, 0x3C, 0x40, 0x40, 0x20, 0x7C,// u
0x00, 0x1C, 0x20, 0x40, 0x20, 0x1C,// v
0x00, 0x3C, 0x40, 0x30, 0x40, 0x3C,// w
0x00, 0x44, 0x28, 0x10, 0x28, 0x44,// x
0x00, 0x1C, 0xA0, 0xA0, 0xA0, 0x7C,// y
0x00, 0x44, 0x64, 0x54, 0x4C, 0x44,// z
0x14, 0x14, 0x14, 0x14, 0x14, 0x14,// horiz lines

};
/****************************************8*16 的 点 阵
**********************************/

const unsigned char F8X16[]=
{


0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,//
0


0x00,0x00,0x00,0xF8,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x33,0x30,0x00,0x00,0x00,//!
```

1

0x00,0x10,0x0C,0x06,0x10,0x0C,0x06,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,/
/" 2

0x40,0xC0,0x78,0x40,0xC0,0x78,0x40,0x00,0x04,0x3F,0x04,0x04,0x3F,0x04,0x04,0x00,/
/# 3

0x00,0x70,0x88,0xFC,0x08,0x30,0x00,0x00,0x00,0x18,0x20,0xFF,0x21,0x1E,0x00,0x00,
//$ 4

0xF0,0x08,0xF0,0x00,0xE0,0x18,0x00,0x00,0x00,0x21,0x1C,0x03,0x1E,0x21,0x1E,0x00,
//% 5

0x00,0xF0,0x08,0x88,0x70,0x00,0x00,0x00,0x1E,0x21,0x23,0x24,0x19,0x27,0x21,0x10,/
/& 6

0x10,0x16,0x0E,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,/
/' 7

0x00,0x00,0x00,0xE0,0x18,0x04,0x02,0x00,0x00,0x00,0x00,0x07,0x18,0x20,0x40,0x00,/
/(8

0x00,0x02,0x04,0x18,0xE0,0x00,0x00,0x00,0x00,0x40,0x20,0x18,0x07,0x00,0x00,0x00,/
/) 9

0x40,0x40,0x80,0xF0,0x80,0x40,0x40,0x00,0x02,0x02,0x01,0x0F,0x01,0x02,0x02,0x00,//
* 10

0x00,0x00,0x00,0xF0,0x00,0x00,0x00,0x00,0x01,0x01,0x01,0x1F,0x01,0x01,0x01,0x00,//
+ 11

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x80,0xB0,0x70,0x00,0x00,0x00,0x00,0x00,/
/, 12

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x01,0x01,0x01,0x01,0x01,0x01,//
- 13

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x30,0x00,0x00,0x00,0x00,0x00,//.
14

0x00,0x00,0x00,0x00,0x80,0x60,0x18,0x04,0x00,0x60,0x18,0x06,0x01,0x00,0x00,0x00,//
/ 15

0x00,0xE0,0x10,0x08,0x08,0x10,0xE0,0x00,0x00,0x0F,0x10,0x20,0x20,0x10,0x0F,0x00,/
/0 16

0x00,0x10,0x10,0xF8,0x00,0x00,0x00,0x00,0x00,0x20,0x20,0x3F,0x20,0x20,0x00,0x00,//
1 17

0x00,0x70,0x08,0x08,0x08,0x88,0x70,0x00,0x00,0x30,0x28,0x24,0x22,0x21,0x30,0x00,//
2 18

0x00,0x30,0x08,0x88,0x88,0x48,0x30,0x00,0x00,0x18,0x20,0x20,0x20,0x11,0x0E,0x00,/
/3 19

0x00,0x00,0xC0,0x20,0x10,0xF8,0x00,0x00,0x00,0x07,0x04,0x24,0x24,0x3F,0x24,0x00,/
/4 20

0x00,0xF8,0x08,0x88,0x88,0x08,0x08,0x00,0x00,0x19,0x21,0x20,0x20,0x11,0x0E,0x00,/
/5 21

0x00,0xE0,0x10,0x88,0x88,0x18,0x00,0x00,0x00,0x0F,0x11,0x20,0x20,0x11,0x0E,0x00,//6 22

0x00,0x38,0x08,0x08,0xC8,0x38,0x08,0x00,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x00,//7 23

0x00,0x70,0x88,0x08,0x08,0x88,0x70,0x00,0x00,0x1C,0x22,0x21,0x21,0x22,0x1C,0x00,//8 24

0x00,0xE0,0x10,0x08,0x08,0x10,0xE0,0x00,0x00,0x00,0x31,0x22,0x22,0x11,0x0F,0x00,//9 25

0x00,0x00,0x00,0xC0,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x30,0x00,0x00,0x00,//: 26

0x00,0x00,0x00,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x80,0x60,0x00,0x00,0x00,0x00,//; 27

0x00,0x00,0x80,0x40,0x20,0x10,0x08,0x00,0x00,0x01,0x02,0x04,0x08,0x10,0x20,0x00,//< 28

0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x00,0x04,0x04,0x04,0x04,0x04,0x04,0x04,0x00,//= 29

0x00,0x08,0x10,0x20,0x40,0x80,0x00,0x00,0x00,0x20,0x10,0x08,0x04,0x02,0x01,0x00,//> 30

0x00,0x70,0x48,0x08,0x08,0x08,0xF0,0x00,0x00,0x00,0x00,0x30,0x36,0x01,0x00,0x00,//?31

0xC0,0x30,0xC8,0x28,0xE8,0x10,0xE0,0x00,0x07,0x18,0x27,0x24,0x23,0x14,0x0B,0x00

,//@ 32

0x00,0x00,0xC0,0x38,0xE0,0x00,0x00,0x00,0x20,0x3C,0x23,0x02,0x02,0x27,0x38,0x20,
//A 33

0x08,0xF8,0x88,0x88,0x88,0x70,0x00,0x00,0x20,0x3F,0x20,0x20,0x20,0x11,0x0E,0x00,/
/B 34

0xC0,0x30,0x08,0x08,0x08,0x08,0x38,0x00,0x07,0x18,0x20,0x20,0x20,0x10,0x08,0x00,/
/C 35

0x08,0xF8,0x08,0x08,0x08,0x10,0xE0,0x00,0x20,0x3F,0x20,0x20,0x20,0x10,0x0F,0x00,/
/D 36

0x08,0xF8,0x88,0x88,0xE8,0x08,0x10,0x00,0x20,0x3F,0x20,0x20,0x23,0x20,0x18,0x00,/
/E 37

0x08,0xF8,0x88,0x88,0xE8,0x08,0x10,0x00,0x20,0x3F,0x20,0x00,0x03,0x00,0x00,0x00,/
/F 38

0xC0,0x30,0x08,0x08,0x08,0x38,0x00,0x00,0x07,0x18,0x20,0x20,0x22,0x1E,0x02,0x00,/
/G 39

0x08,0xF8,0x08,0x00,0x00,0x08,0xF8,0x08,0x20,0x3F,0x21,0x01,0x01,0x21,0x3F,0x20,/
/H 40

0x00,0x08,0x08,0xF8,0x08,0x08,0x00,0x00,0x00,0x20,0x20,0x3F,0x20,0x20,0x00,0x00,//
I 41

0x00,0x00,0x08,0x08,0xF8,0x08,0x08,0x00,0xC0,0x80,0x80,0x80,0x7F,0x00,0x00,0x00,/
/J 42

0x08,0xF8,0x88,0xC0,0x28,0x18,0x08,0x00,0x20,0x3F,0x20,0x01,0x26,0x38,0x20,0x00,//K 43

0x08,0xF8,0x08,0x00,0x00,0x00,0x00,0x00,0x20,0x3F,0x20,0x20,0x20,0x20,0x30,0x00,//L 44

0x08,0xF8,0xF8,0x00,0xF8,0xF8,0x08,0x00,0x20,0x3F,0x00,0x3F,0x00,0x3F,0x20,0x00,//M 45

0x08,0xF8,0x30,0xC0,0x00,0x08,0xF8,0x08,0x20,0x3F,0x20,0x00,0x07,0x18,0x3F,0x00,//N 46

0xE0,0x10,0x08,0x08,0x08,0x10,0xE0,0x00,0x0F,0x10,0x20,0x20,0x20,0x10,0x0F,0x00,//O 47

0x08,0xF8,0x08,0x08,0x08,0x08,0xF0,0x00,0x20,0x3F,0x21,0x01,0x01,0x01,0x00,0x00,//P 48

0xE0,0x10,0x08,0x08,0x08,0x10,0xE0,0x00,0x0F,0x18,0x24,0x24,0x38,0x50,0x4F,0x00,//Q 49

0x08,0xF8,0x88,0x88,0x88,0x88,0x70,0x00,0x20,0x3F,0x20,0x00,0x03,0x0C,0x30,0x20,//R 50

0x00,0x70,0x88,0x08,0x08,0x08,0x38,0x00,0x00,0x38,0x20,0x21,0x21,0x22,0x1C,0x00,//S 51

0x18,0x08,0x08,0xF8,0x08,0x08,0x18,0x00,0x00,0x00,0x20,0x3F,0x20,0x00,0x00,0x00,//T 52

0x08,0xF8,0x08,0x00,0x00,0x08,0xF8,0x08,0x00,0x1F,0x20,0x20,0x20,0x20,0x1F,0x00,//U 53

0x08,0x78,0x88,0x00,0x00,0xC8,0x38,0x08,0x00,0x00,0x07,0x38,0x0E,0x01,0x00,0x00,//V 54

0xF8,0x08,0x00,0xF8,0x00,0x08,0xF8,0x00,0x03,0x3C,0x07,0x00,0x07,0x3C,0x03,0x00,//W 55

0x08,0x18,0x68,0x80,0x80,0x68,0x18,0x08,0x20,0x30,0x2C,0x03,0x03,0x2C,0x30,0x20,//X 56

0x08,0x38,0xC8,0x00,0xC8,0x38,0x08,0x00,0x00,0x00,0x20,0x3F,0x20,0x00,0x00,0x00,//Y 57

0x10,0x08,0x08,0x08,0xC8,0x38,0x08,0x00,0x20,0x38,0x26,0x21,0x20,0x20,0x18,0x00,//Z 58

0x00,0x00,0x00,0xFE,0x02,0x02,0x02,0x00,0x00,0x00,0x00,0x7F,0x40,0x40,0x40,0x00,//[59

0x00,0x0C,0x30,0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x06,0x38,0xC0,0x00,//\ 60

0x00,0x02,0x02,0x02,0xFE,0x00,0x00,0x00,0x00,0x40,0x40,0x40,0x7F,0x00,0x00,0x00,//] 61

0x00,0x00,0x04,0x02,0x02,0x02,0x04,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,//^ 62

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,//

_ 63

0x00,0x02,0x02,0x04,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,//
` 64

0x00,0x00,0x80,0x80,0x80,0x80,0x00,0x00,0x00,0x19,0x24,0x22,0x22,0x22,0x3F,0x20,//
a 65

0x08,0xF8,0x00,0x80,0x80,0x00,0x00,0x00,0x00,0x3F,0x11,0x20,0x20,0x11,0x0E,0x00,/
/b 66

0x00,0x00,0x00,0x80,0x80,0x80,0x00,0x00,0x00,0x0E,0x11,0x20,0x20,0x20,0x11,0x00,//
c 67

0x00,0x00,0x00,0x80,0x80,0x88,0xF8,0x00,0x00,0x0E,0x11,0x20,0x20,0x10,0x3F,0x20,/
/d 68

0x00,0x00,0x80,0x80,0x80,0x80,0x00,0x00,0x00,0x1F,0x22,0x22,0x22,0x22,0x13,0x00,//
e 69

0x00,0x80,0x80,0xF0,0x88,0x88,0x88,0x18,0x00,0x20,0x20,0x3F,0x20,0x20,0x00,0x00,//
f 70

0x00,0x00,0x80,0x80,0x80,0x80,0x80,0x00,0x00,0x6B,0x94,0x94,0x94,0x93,0x60,0x00,/
/g 71

0x08,0xF8,0x00,0x80,0x80,0x80,0x00,0x00,0x20,0x3F,0x21,0x00,0x00,0x20,0x3F,0x20,//
h 72

0x00,0x80,0x98,0x98,0x00,0x00,0x00,0x00,0x00,0x20,0x20,0x3F,0x20,0x20,0x00,0x00,//
i 73

0x00,0x00,0x00,0x80,0x98,0x98,0x00,0x00,0x00,0xC0,0x80,0x80,0x80,0x7F,0x00,0x00,//j 74

0x08,0xF8,0x00,0x00,0x80,0x80,0x80,0x00,0x20,0x3F,0x24,0x02,0x2D,0x30,0x20,0x00,//k 75

0x00,0x08,0x08,0xF8,0x00,0x00,0x00,0x00,0x00,0x20,0x20,0x3F,0x20,0x20,0x00,0x00,//l 76

0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x00,0x20,0x3F,0x20,0x00,0x3F,0x20,0x00,0x3F,//m 77

0x80,0x80,0x00,0x80,0x80,0x80,0x00,0x00,0x20,0x3F,0x21,0x00,0x00,0x20,0x3F,0x20,//n 78

0x00,0x00,0x80,0x80,0x80,0x80,0x00,0x00,0x00,0x1F,0x20,0x20,0x20,0x20,0x1F,0x00,//o 79

0x80,0x80,0x00,0x80,0x80,0x00,0x00,0x00,0x80,0xFF,0xA1,0x20,0x20,0x11,0x0E,0x00,//p 80

0x00,0x00,0x00,0x80,0x80,0x80,0x80,0x00,0x00,0x0E,0x11,0x20,0x20,0xA0,0xFF,0x80,//q 81

0x80,0x80,0x80,0x00,0x80,0x80,0x80,0x00,0x20,0x20,0x3F,0x21,0x20,0x00,0x01,0x00,//r 82

0x00,0x00,0x80,0x80,0x80,0x80,0x80,0x00,0x00,0x33,0x24,0x24,0x24,0x24,0x19,0x00,//s 83

0x00,0x80,0x80,0xE0,0x80,0x80,0x00,0x00,0x00,0x00,0x00,0x1F,0x20,0x20,0x00,0x00,//t 84

0x80,0x80,0x00,0x00,0x00,0x80,0x80,0x00,0x00,0x1F,0x20,0x20,0x20,0x10,0x3F,0x20,//u 85

0x80,0x80,0x80,0x00,0x00,0x80,0x80,0x80,0x00,0x01,0x0E,0x30,0x08,0x06,0x01,0x00,//v 86

0x80,0x80,0x00,0x80,0x00,0x80,0x80,0x80,0x0F,0x30,0x0C,0x03,0x0C,0x30,0x0F,0x00,//w 87

0x00,0x80,0x80,0x00,0x80,0x80,0x80,0x00,0x00,0x20,0x31,0x2E,0x0E,0x31,0x20,0x00,//x 88

0x80,0x80,0x80,0x00,0x00,0x80,0x80,0x80,0x80,0x81,0x8E,0x70,0x18,0x06,0x01,0x00,//y 89

0x00,0x80,0x80,0x80,0x80,0x80,0x80,0x00,0x00,0x21,0x30,0x2C,0x22,0x21,0x30,0x00,//z 90

0x00,0x00,0x00,0x00,0x80,0x7C,0x02,0x02,0x00,0x00,0x00,0x00,0x00,0x3F,0x40,0x40,//{ 91

0x00,0x00,0x00,0x00,0xFF,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0x00,0x00,0x00,//| 92

0x00,0x02,0x02,0x7C,0x80,0x00,0x00,0x00,0x00,0x40,0x40,0x3F,0x00,0x00,0x00,0x00,//} 93

0x00,0x06,0x01,0x01,0x02,0x02,0x04,0x04,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,//

~ 94

};

const char Hzk[][32]={

{0x00,0x20,0x18,0xC7,0x44,0x44,0x44,0x44,0xFC,0x44,0x44,0x44,0x44,0x04,0x00,0x00},
{0x04,0x04,0x04,0x07,0x04,0x04,0x04,0x04,0xFF,0x04,0x04,0x04,0x04,0x04,0x04,0x00},/*"年",0*/

{0x00,0x00,0x00,0xFE,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0xFE,0x00,0x00,0x00},
{0x80,0x40,0x30,0x0F,0x02,0x02,0x02,0x02,0x02,0x02,0x42,0x82,0x7F,0x00,0x00,0x00},/*"月",1*/

{0x00,0x00,0x00,0xFE,0x82,0x82,0x82,0x82,0x82,0x82,0x82,0xFE,0x00,0x00,0x00,0x00},
{0x00,0x00,0x00,0xFF,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0xFF,0x00,0x00,0x00,0x00},/*"日",2*/

};

const unsigned char pic[]={

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x7F,0x7F,0x39,0xB0,0xB0,0xB6,0x96,0xD6,0xD
6,
0xD6,0xD6,0xD6,0xD4,0xB5,0xB1,0xBF,0x3F,0x3F,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x
FF,
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
0xFF,0x7F,0x7F,0x7F,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x00,0xD9,0x3C,0x9F,0x7F,
0x3F,0xC7,0x33,0x5B,0x5B,0xF9,0xE3,0xEF,0xCF,0xB3,0xFB,0x67,0x3F,0xDF,0xFF,0x
20,
0x8F,0xFF,0xFF,0xFF,0x7F,0x3F,0x9F,0xDF,0x0F,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xE3,0x89,0x3D,0x79,0x7B,0xE3,0x67,0xAF,0x2F,0xA
F,
0xB7,0xD7,0xD7,0xD7,0xD7,0xD3,0xD8,0xC6,0xBF,0x3E,0x7C,0xE3,0x0F,0x7F,0xFF,0
xFF,
0xF1,0xF4,0xCF,0xF0,0xE7,0xC4,0xB1,0xEF,0x1E,0xEC,0x3E,0xDB,0xDB,0x7B,0x3B,
0xFF,
0x77,0x01,0x6E,0xCE,0xC3,0xFA,0xFD,0xFF,0x81,0xBC,0x3C,0x8F,0x63,0xB8,0x1E,0x
47,
0x37,0xB3,0x99,0xC5,0xF1,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF
,
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x3F,0xB8,
0x03,0xDF,0xDF,0xD8,0xD2,0x73,0xDB,0xA3,0x5F,0x41,0x5D,0x99,0x13,0xF7,0x30,0x
FD,
0x92,0x3D,0xBF,0x70,0x8F,0xBF,0xBF,0xBF,0x1F,0xCF,0xE9,0xAC,0x26,0x99,0xCA,0x
F4,
0x17,0xF4,0xFF,0xAB,0xA9,0xEE,0xCD,0xB7,0x3E,0x81,0xBF,0xBF,0xBF,0x3F,0x7F,0
x7F,

0x9F,0x8E,0x86,0xA6,0xA2,0xD3,0xD9,0xCD,0xE5,0xF1,0xFF,0xFF,0xFF,0x7F,0x7F,0x7F,
0x7F,0x7F,0x7F,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFE,0xFE,0xFE,0xC0,0x3F,0xFF,0xDF,0x00,0x7B,0xFB,0xC9,0x6F,
0xDE,0x5C,0x1A,0x43,0xEC,0xEF,0x33,0xF8,0x67,0x3F,0xFC,0x02,0xFE,0xC6,0x3A,0xCC,
0xE3,0xF8,0xFF,0xFF,0x7F,0x8F,0xF2,0x18,0xE7,0xF9,0xFC,0xFD,0xFE,0xFE,0xFE,0xFE,
0xFE,0x1E,0xCE,0xFE,0xFE,0xFE,0xFE,0xFE,0xFC,0xFD,0xFC,0xFE,0x1E,0xDE,0xFE,0xFE,
0xFE,0xFE,0xFE,0xFE,0xFF,0x3F,0x9F,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x7F,0xFF,0xBF,0x87,0xF8,0x9F,0xEB,
0x15,0xF7,0x0A,0xA9,0xA3,0x3C,0x77,0x74,0xBF,0x6C,0x3E,0x66,0x9B,0x6D,0x8E,0xDD,
0x7B,0x9A,0xCA,0xF4,0x7F,0x3F,0x9F,0xCF,0xE7,0xF1,0x7C,0x7B,0x1C,0x67,0x7F,0x7F,
0x7F,0x7F,0x7F,0x7F,0x7F,0x7F,0x7F,0x7E,0x7C,0xF9,0xF3,0xF7,0xFF,0xFF,0xFF,0xFF,
0xFF,0x7F,0x7F,0x7C,0x79,0x7B,0x7B,0x7F,0x7F,0x7F,0x7F,0xFC,0xF9,0xF3,0xF7,0xFF,
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xD1,0xAE,0xAB,
0xAD,0xAE,0xAE,0xAE,0x71,0xDE,0xCF,0x57,0x9A,0xD6,0x56,0xB6,0xC7,0x5F,0x3D,0xB5,
0xD4,0xC2,0x7F,0x6D,0x6D,0x5A,0x5F,0x55,0x55,0x56,0x5A,0xAB,0xAB,0xAB,0xA9,0xAF,
0xAD,0xAD,0xAD,0xAD,0xAD,0xAD,0xAD,0xAD,0xAD,0xFD,0x5D,0x5D,0x5D,0x5D,0xDB,0xDA,
0x5A,0x5A,0x5A,0x5A,0xDA,0xDA,0xDA,0xDA,0xDB,0xDB,0x9B,0xBB,0x99,0xD9,0

xDD,0x9D,
0xFC,0xFE,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF
,
0xFF,0xFF,0xFF,0xFF,0xFF,0xD7,0xC7,0xD7,0xD7,0x47,0xF6,0xC0,0x3F,0x7E,0x75,0x7
3,
0xF6,0xFF,0xB2,0x97,0x57,0xD7,0xD7,0xD7,0xD7,0xD7,0xD7,0xD7,0xD7,0xD7,0xD7,
0xD7,
0xD7,0xD7,0xD7,0xD7,0xD7,0xD7,0xD7,0xD7,0xD7,0xD7,0xD7,0xD7,0xD7,0xD7,0xD
7,0xD7,
0xD7,0xD7,0xD7,0xD6,0xDE,0xCE,0xC6,0xD7,0xD6,0xD6,0xD6,0xD6,0xD6,0xD6,0xD
6,0xD6,
0xD6,0xD6,0xD6,0xD7,0xD7,0xD7,0xD7,0xF7,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x
FF,

};
//能天使

const unsigned char pic2[]={

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x3F,0xBF,0xDF,0xDF,0xCF,0xEF,0xEF,0xEF,0xF
7,
0xF7,0xF7,0xF7,0xF7,0xF7,0xF7,0xF7,0xDB,0xDB,0xD7,0xF7,0xF7,0xF7,0xF7,0xF7,0x
F7,
0xEF,0xCF,0xDF,0xDF,0xBF,0xBF,0x7F,0x7F,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF
F,
0xFF,0xFF,0xFF,0xFF,0xFF,0x7F,0x0F,0xE7,0xB8,0x77,0x77,0x37,0xBB,0xA7,0xD9,0x
D7,
0x67,0xBF,0xF7,0x07,0xD7,0xD7,0x5B,0x9F,0xE7,0x7F,0x07,0x7F,0x3F,0x3F,0xFF,0x3F
,

0xBF,0xBF,0xBE,0xBD,0x9B,0xC7,0x6D,0xE3,0xFB,0x7E,0x1C,0xF9,0xF3,0xF7,0x0F,0xFF,
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x80,0x7F,0xFE,0x02,0x7E,0x8F,0x78,0xBF,0xBF,
0xBF,0x8B,0xF4,0x7D,0x81,0x3E,0x7F,0x0F,0xD0,0xFB,0xB2,0xDF,0xDF,0xDF,0xEF,0xEC,
0xEE,0xEC,0xEE,0xEC,0xFF,0xF1,0xF5,0xEF,0x18,0xCB,0xFA,0xBE,0x99,0xE3,0xF6,0xFE,
0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF8,0xE3,0x0C,0xFF,0x15,0xB5,
0xE5,0xAB,0x8B,0xB3,0x37,0x67,0x67,0xF7,0x77,0x67,0xCF,0xE7,0xF7,0x57,0x7B,0xAB,
0x6B,0xAB,0xAB,0x6B,0x7B,0x5B,0xD7,0xC7,0xFF,0xFF,0xFF,0xF8,0x07,0xFF,0x7F,0xBF,
0xFF,0x07,0xF8,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x00,0xFF,
0xFE,0xFF,0xF0,0xF6,0x36,0xB2,0x3B,0xFC,0xFF,0xBF,0x87,0x78,0xFC,0xC5,0xBD,0x35,
0xED,0xED,0xEC,0xEE,0xED,0xED,0xE5,0xFC,0xFD,0xFF,0xFF,0x7F,0xFF,0xFF,0x30,0x77,
0xF4,0x7F,0x83,0xFC,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
0x70,0x0F,0xFF,0xFF,0xFF,0xFF,0x5F,0x6F,0x68,0x2B,0x9D,0x75,0x7B,0x6A,0xEB,0xEB,
0xEF,0xF2,0xFC,0xFB,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x7F,0xB1,0xDE,0x7F,0xA4,
0x97,0x33,0x79,0xFC,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xCF,0xEF,0xF7,0xF9,
0xFD,0xFD,0x80,0x7F,0xEF,0xFC,0xE1,0x1F,0xBD,0x3D,0xFC,0xFD,0xFD,0xFD,0xFD,0xFB,
0xFA,0xFA,0xFA,0xFA,0xF9,0xFF,0xFF,0xFF,0x7F,0x7F,0xBF,0xDF,0xDF,0xE7,0xF3,0xF9,

0xFE,0xE3,0x3F,0x87,0xF8,0xFD,0xF9,0xF3,0xF7,0xE7,0xEF,0xDF,0x9F,0x3F,0x0F,0x0F,

0x0F,0x0F,0x0F,0x0F,0x0F,0x0E,0x08,0x0B,0x0F,0x0C,0x03,0x0F,0x08,0x0F,0x0F,0x0F,

0x0F,0x0F,0x0F,0x0F,0x0F,0x0F,0x0D,0x0D,0x0E,0x0E,0x0F,0x0F,0x0F,0x0F,0x0F,0x0F,

0x0F,0x0F,0x03,0x08,0x0F,0x0F,0x0F,0x0F,0x0F,0x0F,0x0F,0x0F,0x0F,0x0F,0x0F,0x02,

//诚哥

};

#endif