ESP32控制显示器数码管

安装库

TM1637驱动4位数码管（4 digital display）是一种比较常用的模块，也是比较常见的数码管驱动方式。为了在数码管上显示信息，尝试了库TM1637（不能用），库[Grove\_4Digital\_Display](https://github.com/Seeed-Studio/Grove_4Digital_Display)（好用），因此建议安装后者。

测试中采用了GPIO2作为数码管模块的CLK，GPIO3作为数码管模块的DIO，启用了数码管上的两个点。

tips：以下的代码段的取余很有意思，否则会显示“ABCDEF”和息屏。

tm1637.display(0, (numCounter / 1000%10));

tm1637.display(1, (numCounter / 100%10));

tm1637.display(2, (numCounter / 10%10));

tm1637.display(3, (numCounter % 10));

2.库中提供了三个示例

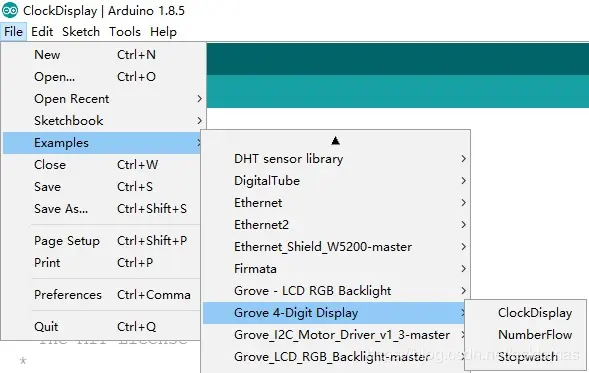
分别为：ClockDisplay、NumberFlow、Stopwatch，其中：

ClockDisplay：时钟显示示例，显示小时和分钟。

NumberFlow：循环显示数组内容示例

Stopwatch：这个是个秒表的示例

这里使用ClockDisplay



图片包含 游戏机, 桌子, 钟表

描述已自动生成

过程就不多说了，直接上结论。在某个位置上显示某个值的代码为：

digitalWrite(rck, LOW);

shiftOut(dio, sck, MSBFIRST, 数字);

shiftOut(dio, sck, MSBFIRST, 位置);

digitalWrite(rck, HIGH);

因为是两片74HC595做级联，所以相当于变成了16位的移位寄存。其中，前8位用来显示7段码（和小数点），后8位用来指示由哪个位置上的数码单元显示。

DIO相当于位输入，RCK相当于各位输入确认的时钟，SCK相当于整体输入确认的时钟。

对于每个数码单元，a-g的排列是：8字中间那一横是a，然后从左上那一竖开始顺时针画圈数，分别是b-g。DIO输入次序是：第一位是小数点，第二位到第八位是a-g。需要亮的段写1，反之则写0。

所以0-9十个数字的二进制值是：

byte num[10] = {

B00111111, B00000110, B01011011, B01001111, B01100110,

B01101101, B01111101, B00000111, B01111111, B01101111

};

小数点和负号分别是：

byte dot = B10000000;

byte neg = B01000000;

对于位选，规律是：从左到右，选中第一位的代码是 B11111110，第二位是B11111101，以此类推。所以位置代码为：

byte pos[4] = {0xFE, 0xFD, 0xFB, 0xF7};

该显示模块有5个引脚，分别是：VCC、SCLK、RCLK、DIO、GND，其中VCC接5V电源，GND接地，SCLK用来接收同步时钟，RCLK用来控制数据的输出，DIO用来接收串行数据的输入。不过我们只要将SCLK理解为接收7段数码管的段码，RCLK理解为接收数码管的编号或称作位码（即第几个数码管）。如下图所示：



  串行4位数码显示模块与之前直接连接4位数码管的显示原理相同，也是采用“动态扫描显示”的方法进行显示，即依次显示第1位数码管、第2位数码管、第3位数码管、第4位数码管，此方法是利用人眼的视觉残留，当扫描显示的频率大于25帧/秒，我们看到的就是每个数码管同时都在显示。同样为解决在主循环中进行显示处理，有可能造成显示不正常情况，我们还是采用Arduino的时钟中断来进行“动态扫描显示”，时钟中断间隔定为5ms。

    本次实验采用Arduino UNO[开发板](https://so.csdn.net/so/search?q=%E5%BC%80%E5%8F%91%E6%9D%BF&spm=1001.2101.3001.7020)，显示模块的SCLK、RCLK、DIO引脚依次接到Arduino的D2、D3、D4，

    程序在主循环中间隔0.1秒对静态变量num进行累加，然后将num值显示到串行4位数码显示模块上。完整程序（该程序经过测试）如下：

#define SCLK 2 //串行时钟引脚定义

#define RCLK 3 //接收时钟引脚定义

#define DIO 4 //数据引脚定义

#include<MsTimer2.h>      //调用定时器T2的库文件

/\*

  采用时钟中断方式，串行4位数码模块动态显示0—9999程序

  优点：可在主循环中任意使用delay 延时函数

\*/

int segCount=4;//4位数码管

//定义段码，这里是共阳段码，可以使用数码管段码软件改变数组值或者任意显示形状

const unsigned char dofly\_DuanMa[]={// 0   1    2    3  4  5    6    7  8    9    A    b  C    d    E    F    -

  0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x98,0x8C,0xBF,0xC6,0xA1,0x86,0x8E,0xbf};

//位码

int segPins[] = {

  0x08,0x04,0x02,0x01};   //对应数码管的编号即位码，从左往右为第1个数码管，第2个数码管、第3个数码管，第4个数码管

unsigned char displayTemp[4];//显示缓冲区，存放4位要显示的段码

void setup() {

// 把对应的端口都设置成输出

  pinMode(SCLK,OUTPUT);

  pinMode(RCLK,OUTPUT);

  pinMode(DIO,OUTPUT);

  MsTimer2::set(5, Timedisp);  //定义Timedisp为时钟中断调用的子程序，每5ms触发时钟中断进入一次Timedisp函数

  MsTimer2::start();      //时钟中断开始

}

// 数据处理，把需要处理的byte数据（段码）写到对应的端口SCLK。

void deal(unsigned char value){

  for(int i=7;i>=0;i--){

    digitalWrite(DIO,bitRead(value,i));//使用了bitWrite函数，非常简单

    // !bitRead(value,i)，这里前面加！(非运算符号)，取决于使用的是共阴还是共阳数码管。

    digitalWrite(SCLK,LOW);

    digitalWrite(SCLK,HIGH);

  }

}

// 时钟中断调用的显示程序

//每中断一次，显示displayTemp[0]—displayTemp[3]中的一个（即静态变量i）段码值一次

void Timedisp() {

  static int i;

  deal(displayTemp[i]);//将对应的段码值写到对应的数码管

  deal(segPins[i]);//将对应的数码管编号（位码）的段码值写到DIO口

  digitalWrite(RCLK,LOW);//复位RCLK

  digitalWrite(RCLK,HIGH);//触发RCLK，即选择相应的编码号为i的数码管

  i++;

  if(i==segCount) //4位结束后重新循环

    i=0;

}

//显示程序，在4位数码中显示num，num小于10000

void disp(unsigned long num) {

  num=num%10000;//取模运算，防止num超过9999

  displayTemp[0]=dofly\_DuanMa[num/1000];       //将num的个位数段码值送到显示缓存displayTemp[0]中

  displayTemp[1]=dofly\_DuanMa[(num%1000)/100];    //将num的十位数段码值送到显示缓存displayTemp[1]中

  displayTemp[2]=dofly\_DuanMa[((num%1000)%100)/10]; //将num的百位数段码值送到显示缓存displayTemp[2]中

  displayTemp[3]=dofly\_DuanMa[((num%1000)%100)%10]; //将num的千位数段码值送到显示缓存displayTemp[3]中

}

// 主循环

void loop() {

    static unsigned int num;//定义一个数据

    disp(num);//显示累加值num

    num++;

    delay(100); //延时100ms

}