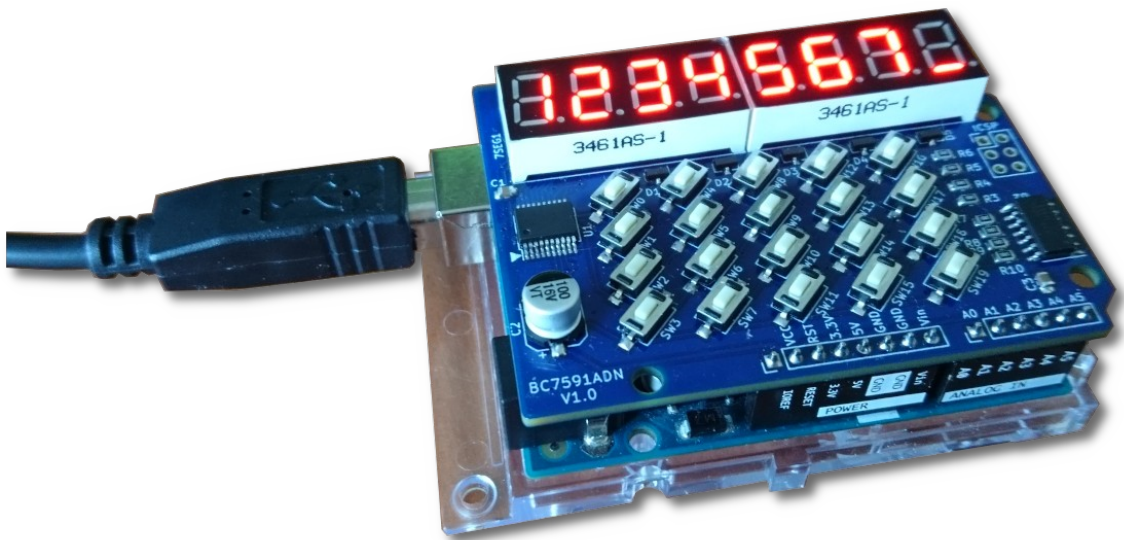


BC7591ADN

8 位数码管及 20 键键盘

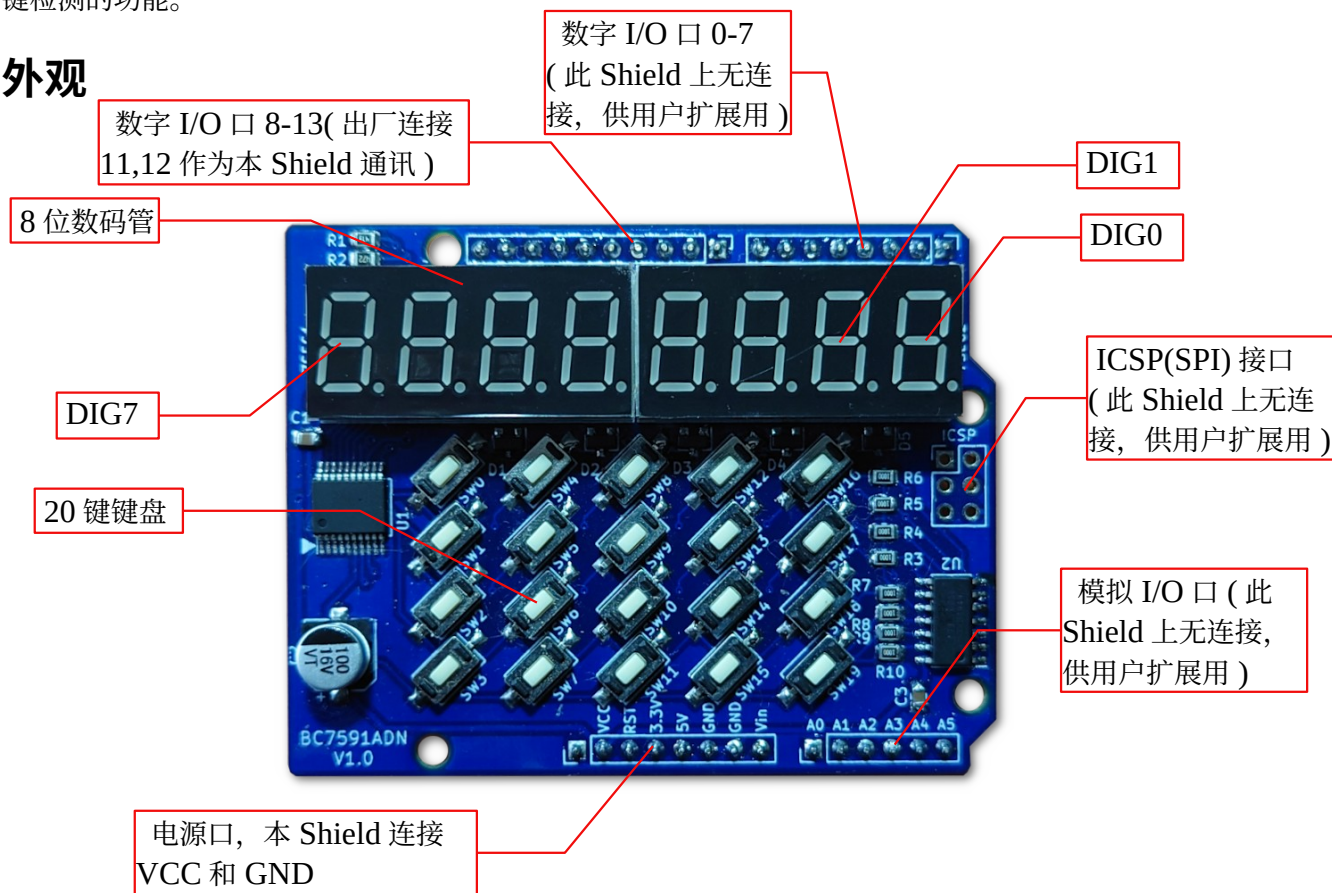
Arduino Shield



- 提供驱动库，Arduino IDE 中直接安装
- 可用于任何 UNO 及 MEGA 尺寸 Arduino 板
- 仅使用两个数字 I/O 口，口线可配置
- 任意显示段均可闪烁显示，闪烁速度可调
- 10 进制和 16 进制显示函数
- 显示亮度可调
- 支持长按键和组合键

BC7591ADN 8 位数码管及 20 键键盘 Arduino Shield 可以提供 8 位的 7 段数码管加小数点的显示功能，外加 20 键的键盘，键盘部分可以具有单个按键按下检测、单个按键释放检测、长按键检测以及组合键检测的功能。

外观

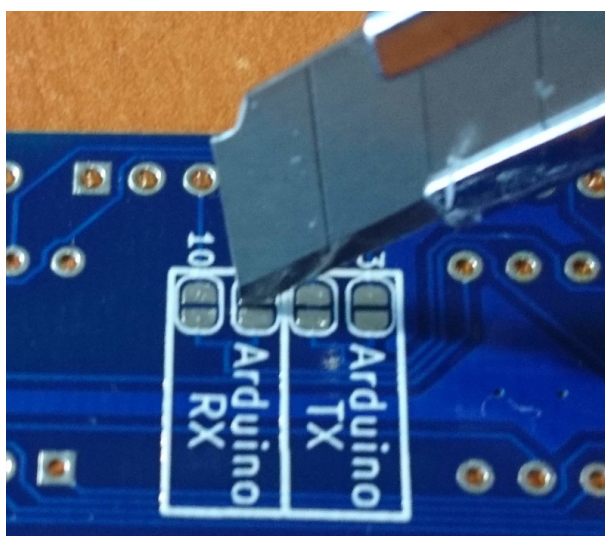


连接接口配置

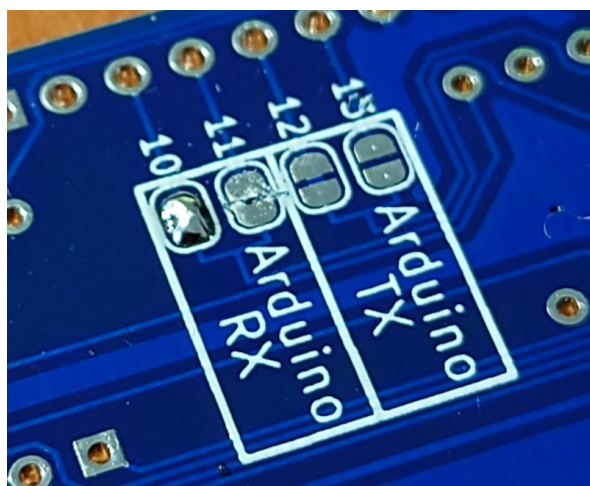
BC7591ADN Shield 基于 BC7591 芯片制作，BC7591 芯片是一款最多具有 32 位数码管和 96 键键盘驱动能力的 LED 显示加键盘接口芯片。

BC7591 芯片采用串口通讯，波特率为 9600，其中 BC7591 的 RX(接 Arduino 的 TX)为显示接口，BC7591 的 TX(接 Arduino 的 RX)为键盘接口。BC7591 可使用 BC6xxx 和 BC759x 专用的 Arduino 驱动库，驱动库分为键盘库和显示驱动库，区分为两个库是因为 BC6xxx 系列芯片为纯键盘芯片，无显示接口，而同时区分单独显示库也可使得 BC759X 芯片做纯显示驱动时程序更精简。驱动库本身支持 Arduino 的硬件串口和软件串口，因为 Arduino UNO 的硬件串口一般作为下载程序使用，因此 BC5791ADN Shield 选择使用 Arduino 的软件串口，连接的数字 I/O 口为用户可选，通过背面的焊接跳线，用户可以选择使用数字 I/O 10 或 11 作为 Arduino 软件串口的 RX，以及 12 或 13 作为软件串口的 TX。

出厂时，电路已经连接数字 I/O 11 和 12 两根线作为与 Arduino 的通讯用，如果用户已经将这个 I/O 用作其它用途，可以切断这两个连线，用跳线方式连接 10 和 13 作为串口。见图：



切断原连接

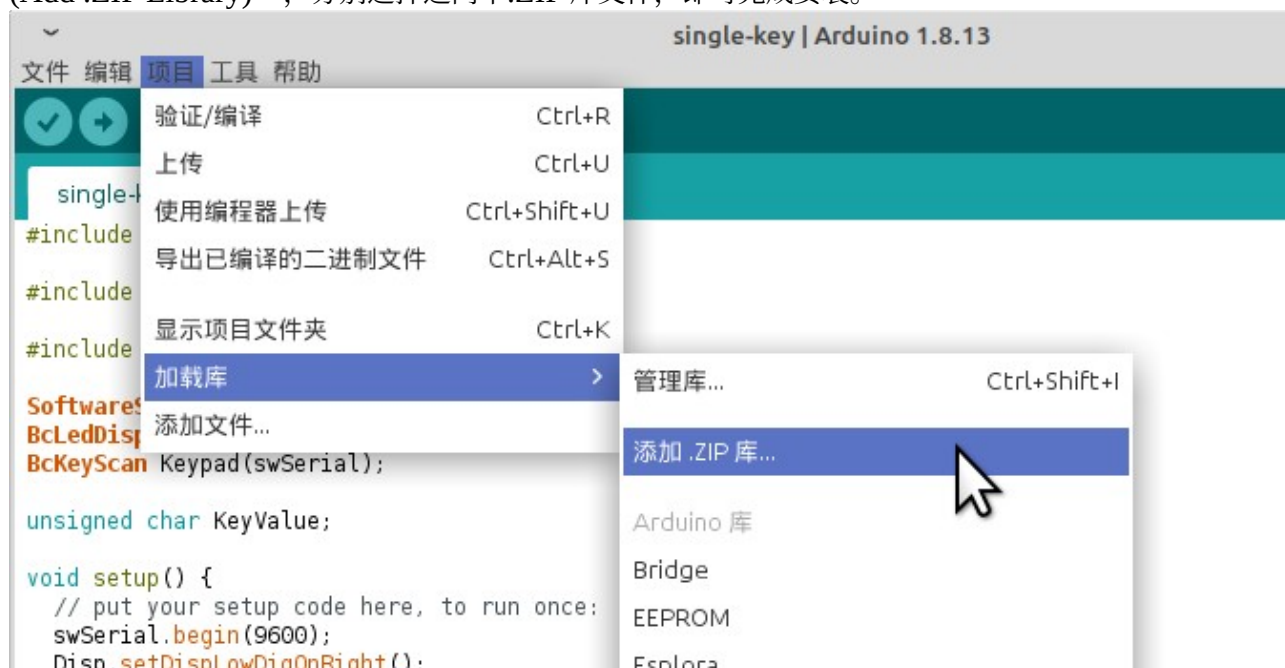


用焊锡连接新的 I/O

请注意修改了硬件连接配置后，例子 Sketch 源程序中的软件部分也须做相应修改。

驱动库安装

驱动库包含两个文件，BC_led_disp.zip 和 BC_key_scan.zip, 分别是显示驱动库和键盘驱动库。安装时，打开 Arduino IDE, 从菜单中选择 “项目(Sketch) -> 加载库(Include Library) -> 添加.ZIP 库...(Add .ZIP Library)”，分别选择这两个.ZIP 库文件，即可完成安装。



两个驱动库的说明书中，有进一步有关手动安装的方法等的说明。

使用

此处讲解从建立新项目开始的使用过程，建议初用者不要从头开始建立新的项目，而是以提供的 4 个例子为样本，在例程的基础上修改程序以符合自己的需求，每次修改一个地方，修改后随时测试运行，这样更容易让程序调试通过。

加载库

安装驱动库后，本 Shield 即可使用。BC7591ADN Shield 使用 Arduino 的软件串口，因此使用时必须同时加载 Arduino SoftwareSerial 库。该库是 Arduino 的标准库之一。打开 Arduino IDE, 在 “项目 -> 加载库” 菜单中，加载下面三个库：

```
SoftwareSerial
BC_key_scan
BC_led_disp
```

建立实例

首先，需要建立软件串口的实例，以供显示和键盘驱动库使用，在#include 语句之后，setup()之前，输入：

```
SoftwareSerial swSerial(11, 12);
```

其中 swSerial 是用户给软件串口实例所起的名字，可以是任何内容，只要符合一般 Arduino 变量名的规则即可。11 和 12 分别是软件串口所使用的数字 I/O 口线，11 为 RX，12 为 TX。这两个值不能随便更改，由本 Shield 的硬件连接决定，出厂时，固定为使用 11 和 12 号数字 I/O，除非对电路板背面的跳线做了修改，这两个值不能变动。

有了软件串口的实例后，就可以建立显示和键盘的实例，在建立软件串口实例的语句下方，输入：

```
BcLedDisp          Disp(swSerial);  
BcKeyScan          Keypad(swSerial);
```

上面两个语句中，swSerial 是上面建立的软件串口的实例的名称，Disp 和 Keypad 分别是显示和键盘的实例的名称，和上面一样，可以由用户任意命名，只要符合 Arduino 变量名的命名规则即可。这两个语句的含义是，显示和键盘都使用 swSerial 软件串口作为通讯口。

初始化

串口必须经初始化后才能使用，因此在 setup() 部分，在任何需要发送数据的显示库操作之前，必须有如下的串口初始化操作：

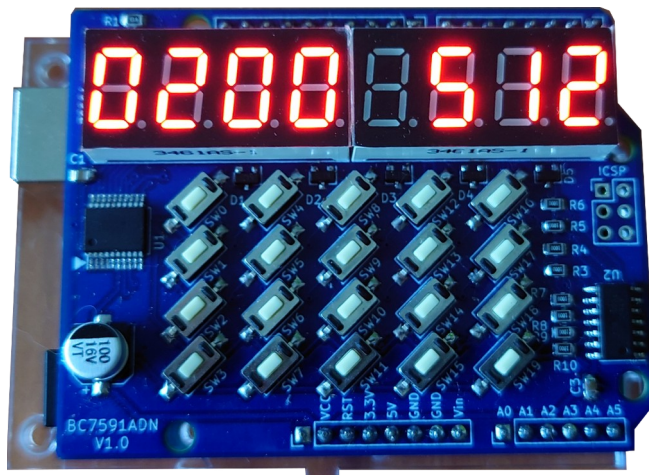
```
swSerial.begin(9600);
```

该操作令串口的参数和 BC7591 芯片所要求的一致。

键盘和显示驱动库本身无需初始化即可工作，驱动库的默认设置即可适合本 Shield 的配置，不过有些操作，比如清屏操作 clear() 等，放在 setup() 中可以确保每次 Arduino 复位后都能回到一个确定的状态。另外，如果要使用键盘库中的长按键和组合键的功能，需要做一些长按键和组合键的定义工作，也可以放在 setup() 中完成。

例程说明

1. 纯显示例子 display-only.ino



该例子未使用键盘功能，仅仅演示显示驱动的功能，因此也仅仅加载了显示驱动库。

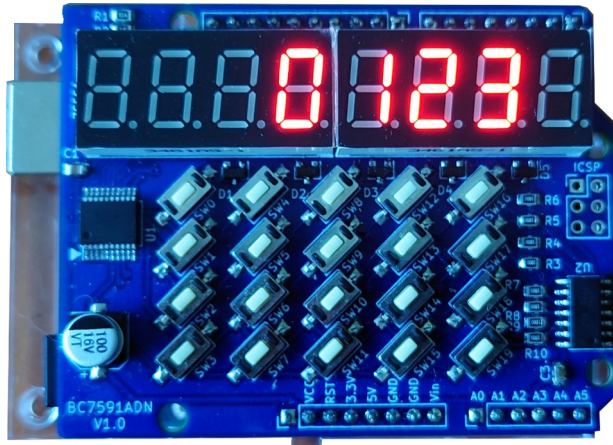
程序开始会显示一个快速计数，计数分成 10 进制和 16 进制两部分同时显示在左右两侧 4 个数码管上，即显示的是同一个数值的 16 进制和 10 进制的方式。计数从 0 开始，直到 0x200 即 10 进制的 512 结束。这部分主要演示驱动库的 16 进制和 10 进制显示函数的功能。

计数结束后，显示一个光点，围绕整个数码管区域做旋转，先按顺时针方向旋转，然后按逆时针方向旋转。这部分程序演示使用 sendCmd() 函数直接发送 BC7591 的段寻址(SEG_ON, SEG_OFF)指令，控制任意显示段的点亮和熄灭。

演示程序第三部分，显示数字“12345.678”，小数点闪烁显示。展示如何使用 sendCmd() 函数控制小数点显示和任意显示段的闪烁。

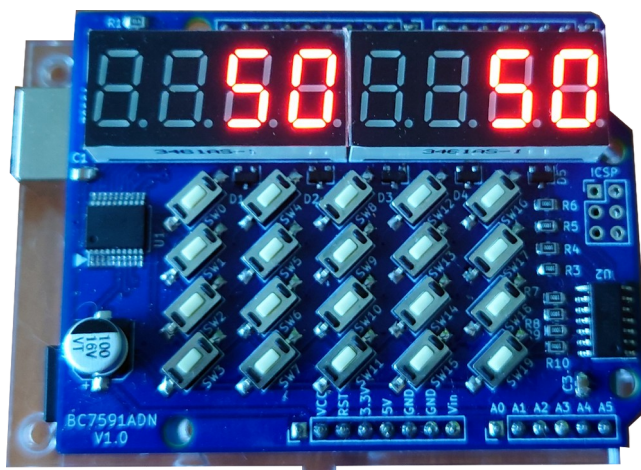
演示程序第四部分，在数码管左侧显示英文“brt=”，brt 三个字母闪烁显示，等号后面一个 16 进制的数值，同时最右侧数码管显示一个两位数的计数。16 进制的数值表示当前的亮度设置值，计数值每计数 50 次，亮度值就变化一次，从 0 变化到 F，然后会再变回到 0。用户会看到数码管显示的亮度逐渐由明到暗再到亮的变化过程。这部分主要演示有特殊字形显示，digitBlink() 函数使用，以及显示亮度控制。

2. 简单按键例子 *single-key.ino*



此例子演示简单按键处理。20 个键键值 0-15 的按键被定义为 16 进制数字 0-F，而最后 4 个按键被定义为 "P", "H", "L", 和空格这几个特殊字符。程序在最右侧数码管闪烁显示一个 "_" 等待用户输入，同时检查按键的情况，如果有按键按下，会把对应的字符显示在数码管上，同时之前的显示内容会向左移，然后等待用户的下一次按键。

3. 全功能键盘例子 *full-feature.ino*



此例子演示了长按键和组合键的使用，包括组合键的长按。程序在数码管左右两侧同时闪烁显示一个 0-100 的十进制数字，初始值是 50. 当按 17 号键(+)键时，右侧的数据加一，当按 18 号键时，右侧的数据减一。这是普通的单按键，如果长按 17 号或者 18 号键 2 秒钟，则右侧数据的闪烁会停止，同时数字开始快速增加或减少，直至按键释放或者达到 100 或 0. 如果按住 3 号键再按 17 或 18 号键，形成组合键，则改为调整左侧的数字，组合键也支持长按，长按组合键则左侧的数字也会停止闪烁快速增加和减少。

这个例子主要演示长按键和组合键的使用。有关长按键和组合键的用法的详细说明，参见键盘驱动库的说明书。

附录 1：驱动库常用函数简要说明

显示驱动库：

clear() - 清除显示

该函数清除一切数码管上显示的内容，并清除所有的闪烁属性。

displayHex(Val, Pos, Width) - 将数值按 16 进制显示

将数值 Val 按 16 进制显示在从第 Pos 位数码管开始的 Width 个数码管上。

displayDec(Val, Pos, Width) - 将数值按 10 进制显示

将数值 Val 按 10 进制显示在从第 Pos 位数码管开始的 Width 个数码管上。

digitBlink(Digit, On/Off) - 数码管按位闪烁

控制某一位数码管(Digit)是否闪烁，On/Off 控制是否闪烁，为 1 时闪烁，0 为不闪烁。

sendCmd(Cmd, Data) - 向 BC7591 发送指令

该命令直接向 BC7591 发送指令，Cmd 是 BC7591 指令，而 Data 是数据。驱动库的所有功能内部均为调用此函数实现，通过此函数可以直接控制 Shield 上的 BC7591 芯片，完成一些驱动库没有提供的特殊功能，比如控制显示亮度，显示内容的左移/右移等。详细的 BC7591 指令说明请参考 BC7591 芯片技术手册。

键盘驱动库：

setDetectMode(Mode) - 设置检测模式

Mode 为 0 时，只检测按键按下，不检测按键释放，为 1 时，同时检测按键按下和释放。

checkChanges() - 更新键盘状态

在 loop() 中，应该定期呼叫 checkChanges()，以便驱动库随时掌握键盘的状态。

isKeyChanged() - 查询是否有新按键

返回值为 true(1) 或者 false(0)，通过这个查询，用户就可以了解是否键盘有新的还未处理的按键。

getKeyValue() - 获取键值

函数返回最近一次的按键键值。这里的按键不仅指单独的物理按键，也包括用户自定义的组合键和长按键。

setLongpressCount(CountLimit) - 设置长按键计数值

驱动库的长按键时间通过一个内部的计数器来实现，计数器达到设定的限值，就被认为达到了长按键的时间，此函数用来设置这个限值。

longpressTick() - 长按键计数

使用长按键时，在 loop() 中定期呼叫这个函数，当达到设定的次数限值时，即认为达到了长按键的时间，设定的长按键一直处在按下的状态，就会产生一个长按键事件。

defLongpressKey() - 定义长按键

使用长按键之前，必须先定义长按键，让驱动库知道哪些键是需要监视长按动作的，以及分配给这些长按键的自定义键值是什么。请参阅键盘驱动库的说明书了解详细使用方法。

defCombinedKey() - 定义组合键

组合键的使用和长按键类似，也需要事先定义哪些键的组合是需要驱动库检测的，并给每个组合键定义一个特殊的键值。具体的用法也请参阅键盘驱动库的说明书。

附录 2：BC7591 常用指令

通过 sendCmd()函数，可以直接控制本 Shield 上的主要芯片 BC7591 的工作，完成很多特定的功能，如控制显示亮度，显示特殊字符等等。完整的控制指令说明和用法，请参阅 BC7591 的技术手册。

DIRECT_WT - 直接写入显示寄存器，主要用来显示特殊字形符号，如 P, H 等。使用格式：

```
sendCmd(DIRECT_WT+pos, data);
```

其中，pos 是写入的显示位，data 是写入的映射数据，1 对应位点亮，0 对应段熄灭。

BLINK_WT_CLR - 段闪烁清除，使用格式：

```
sendCmd(BLINK_WT_CLR+pos, data);
```

其中 pos 是显示位，data 中为 1 的位对应的显示段的闪烁属性会被清除。

BLINK_WT_SET - 段闪烁置位，使用格式：

```
sendCmd(BLINK_WT_SET+pos, data);
```

其中 pos 是显示位，data 中 1 对应的显示段会变为闪烁显示。

SHIFT_H_WT - 向高显示位移动插入，将数据插入到显示寄存器，同时将原显示内容向高显示位方向移动。使用格式：

```
sendCmd(SHIFT_H_WT+pos, data);
```

其中 pos 是显示位，data 是数码管映射数据。

SHIFT_L_WT - 向低显示位移动插入，和上条命令作用类似，不过插入移动方向相反。

SEG_ON - 段点亮，点亮一个特定显示段。使用格式：

```
sendCmd(SEG_ON, seg_number);
```

seg_number 是段地址，第 0 位的 A 段地址为 0, DP 段为 7, 第 1 位的 A 段地址为 8, 依次类推。

SEG_OFF - 段熄灭，关闭一个特定显示段。和上面指令类似，但作用是关闭显示。

WRITE_ALL - 全局写入，数据将被写入所有的显示寄存器。可以用来做点亮全部显示等。使用格式：

```
sendCmd(WRITE_ALL, data);
```

BLINK_SPEED - 闪烁速度控制。闪烁的频率大致可以用 $F_{\text{blink}} = 92/(2*S)$ 计算。使用格式：

```
sendCmd(BLINK_SPEED, S);
```

DIM_CTL - 显示亮度控制。控制值范围为 0-F，为 0 时，亮度最大，为 F 时最暗。使用格式：

```
sendCmd(DIM_CTL, ctl);
```

附录 3：电路原理图

