

BitCode

UART 7 段 LED 显示驱动

Arduino 库

使用说明书

索引

索引.....	2
综述.....	3
驱动库的安装.....	4
硬件连接.....	5
驱动库的使用.....	6
建立实例.....	6
初始化 setup().....	6
显示内容.....	7
显示特殊字符.....	7
显示小数点.....	7
其它特殊控制.....	8
函数参考手册.....	9
sendCmd() 发送指令.....	9
clear() 清除显示和闪烁.....	10
displayDec() 按 10 进制显示数值.....	10
displayHex() 按 16 进制显示数值.....	10
displayFloat() 显示浮点数.....	11
digitBlink() 位闪烁控制.....	11
使用方法示例.....	12

综述

BC759x 系列 7 段 LED 显示驱动+键盘接口芯片提供统一的 UART 单线 LED 显示接口，本驱动库可适用于以下芯片：

BC7595 —— 48 段(6 位数码管)+48 键键盘接口芯片

BC7591 —— 256 段(32 位数码管)+96 键键盘接口芯片

本驱动库适用于所有的 Arduino 器件，可用于硬件串口和软件串口。

BC759x 芯片每个指令由 2 个字节组成，第一个字节为指令，第二个字节为数据。驱动库提供了一个基本的操作函数 `sendCmd()`，可以用来向 BC759x 发送任意指令。同时，驱动库提供了几个上层函数，包装了使用中最常用的几个功能。上层函数如下：

`clear()` - 清除显示和闪烁状态

`displayDec()` - 按 10 进制显示数值

`displayHex()` - 按 16 进制显示数值

`displayFloat()` - 显示浮点数

`digitBlink()` - 按数码管位控制闪烁显示

有些功能虽然很常用——比如单个 LED 的亮/灭控制，但可以用单个 BC759x 指令完成的，也不再另外提供上层函数，因为这样做并不能简化用户的使用。

有关 BC759x 芯片的指令的详细资料，请查阅相关芯片的数据手册。

驱动库的安装

驱动库的安装非常简单，直接从 Arduino 的库管理器中，搜索本库名或芯片名，即可找到本驱动直接安装，或者直接下载 zip 文件后，从 Arduino IDE 菜单中，选择

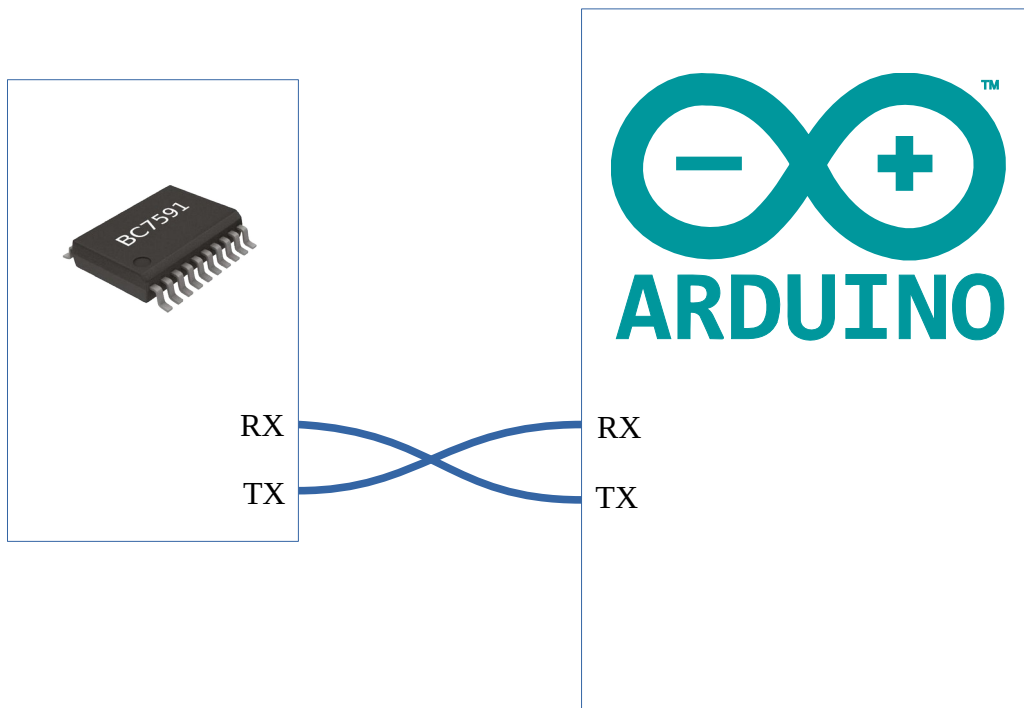
“项目(Sketch) --> 加载库(Include Library) --> 添加.ZIP 库...(Add .ZIP Library...)”

选择 uart_7seg_display.zip 文件，即可完成安装。

安装完成后，即可在“项目 (Sketch) --> 加载库 (Include Library)”菜单中，看到 UART_7seg_Display 库，表示已经可以使用了。

硬件连接

BC759x 芯片采用串口通讯，通讯采用 2 个引脚 TX 和 RX，其中 TX 为 BC759x 芯片的键盘接口的输出，显示部分其实只用到 RX 引脚，如果只使用 BC759x 的显示功能而不使用其键盘接口，则除了电源外，只需要连接 RX 一个引脚就可以。因为通常显示和键盘会一起使用，因此 RX 和 TX 一般都会连接。连接时，芯片的 TX/RX 引脚与 Arduino 的串口 RX/TX 引脚交叉对接：



驱动库的使用

建立实例

BC759x 芯片的接口为串口，因此本驱动库的使用，也必须依赖于 Arduino 的串口。串口可以是 Arduino 的硬件串口(如 Serial, Serial1, Serial2 等)，也可以是软件串口(Software Serial)。

对于像 Arduino UNO 这样的型号，只有一个硬件串口，且该串口已经用于和电脑的通讯，因此适于使用软件串口，否则会 and IDE 发生冲突（除非程序下载到硬件使用外部编程器不占用 Arduino 的串口）。

使用前，第一步应加载驱动库，加载后，Sketch 编辑窗口出现下面语句：

```
#include <UART_7Seg_Display.h>
```

其次，为方便使用，可以定义一个常数 LED_SERIAL 代表所使用的串口

```
#define LED_SERIAL Serial1
```

在上面 include 语句的下面，setup()之前，输入建立本驱动类的实例的语句，如：

```
UART_7Seg_Display Disp(LED_SERIAL);
```

其中 ART_7Seg_Display 是本驱动类的名称，Disp 是用户给实例起的名字，可以是任意内容，只要符合 Arduino 变量命名规则即可。括号内是所使用的串口。

如果使用软件串口，需要 2 个额外的步骤，第一步需要加载软件串口库 SoftwareSerial，该库是 Arduino 的标准库之一。

```
#include <SoftwareSerial.h>
```

第二步需要在建立本驱动的实例之前，先建立一个软件串口的实例：

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

其中 swSerial 是用户给实例所起的名称，可以是符合 Arduino 变量命名规则的任何内容。括号内 11, 12 是软件串口所使用的 RX 和 TX 引脚。详情请参阅 Arduino 软件串口库的资料。

上面 LED_SERIAL 的定义语句，也需做相应改动

```
#define LED_SERIAL swSerial
```

初始化 setup()

本驱动库本身并不需要做任何初始化即可使用，不过串口必须经过正确的初始化，才能正确连接 BC759x 芯片。串口必须初始化为 9600 波特率。

```
LED_SERIAL.begin(9600);
```

虽然无需初始化即可使用，但因为不同 PCB 硬件设计可能数码管排列方式不同，有可能会需要在初始化中设置一下数字显示的方向。如果驱动库的设置显示方向和实际电路板相反，当显示数值的时候，会造成高位和低位的位置颠倒，比如显示 10 进制数字时，个位可能会显示在最左侧。

驱动库有两个函数设置显示方向，分别是：

`setDispLowDigOnLeft()` 和 `setDispLowDigOnRight()`，分别代表电路板上较小标号的数码管是排在左侧还是右侧。用户根据所用 PCB 的设计呼叫相应函数即可，比如：

```
Disp.setDispLowDigOnLeft();           // 通知驱动库较小序号数码管在左侧
```

如果不设置，驱动库默认较小编号的数码管排在右侧。

此外，清除函数 `clear()` 也经常用在初始化部分，用来在复位后清除所有的显示内容和闪烁属性：

```
Disp.clear();
```

显示内容

BC759x 芯片具有内部的缓存功能，显示内容送入后，会一直显示，直到被新的内容替代。因此程序只需在更新显示内容时呼叫驱动库。

驱动库提供了 3 个和显示有关的函数：

按 10 进制显示 —— `displayDec()`

按 16 进制显示 —— `displayHex()`

按浮点数显示 —— `displayFloat()`

按位闪烁 —— `digitBlink()`

关于这四个函数的使用，参阅后文函数参考资料部分。

驱动库仅提供以上 3 个显示有关函数，因为其它常用操作一般可由一个 BC759x 指令完成，封装成函数并不能简化用户程序。下面是一些常用操作：

显示特殊字符

一些特殊字符，比如“L”，“H”，“P”，“-”等，可以由直接写入显示寄存器实现，使用 BC759x 的直接写入寄存器指令 `DIRECT_WT`，比如显示“L”

```
sendCmd(DIRECT_WT|Pos, 0x38);        // Pos 为显示的位置，0x38 为“L”的字符映射。
```

有关 BC759x 的指令详细说明，可以参考 BC759x 系列芯片的数据手册。

显示小数点

数值显示函数无论是 10 进制显示还是 16 进制或浮点显示，均不会影响小数点的显示，要控制小数点的显示，可以直接使用 BC759x 的段寻址指令 `SEG_OFF/SEG_ON`，比如欲点亮第 3 位的小数点：

```
sendCmd(SEG_ON, 0x1f); // 0x1f 是第 3 位数码管小数点的段地址
```

其它特殊控制

任何 BC759x 芯片的控制，都可以通过 sendCmd() 函数来完成，比如控制闪烁速度，调节显示亮度等等，有关 BC759x 芯片的指令的更多资料，请参考 BC759x 芯片的技术手册。

函数参考手册

sendCmd() 发送指令

函数使用格式：

```
sendCmd(Cmd, Data);
```

这个函数可以用来向 BC759x 发送任意指令。函数库仅提供了几个上层的常用显示函数，如果用户需要对 BC759x 进行全面的控制，则需要通过调用此函数。参数有两个，Cmd 为待发送的指令，Data 为待发送的数据。UART_7Seg_Display.h 头文件中列出了所有 BC759x 芯片的指令的定义。关于具体 BC759x 芯片的指令的详细说明，请参阅所选用的 BC759x 芯片的技术手册。函数库所提供的的所有其它函数，均通过调用此函数实现功能。

预定义指令：

以下指令在库中已经预定义好可直接使用：

- DIRECT_WT 直接写入显示寄存器指令
- COL_WRITE 列写入指令（仅 BC7591）
- BLINK_WT_CLR 清除段闪烁指令
- BLINK_WT_SET 设置段闪烁指令
- SHIFT_H_WT 列插入并向高地址（右）平移指令（仅 BC7591）
- ROTATE_R 向右（高地址）循环滚屏（仅 BC7591）
- ROTATE_L 向左（低地址）循环滚屏（仅 BC7591）
- SHIFT_L_WT 列插入并向低地址（左）平移指令（仅 BC7591）
- DECODE_WT 译码显示
- QTR_WT_BOT 底部 1/4 行写入指令（仅 BC7591）
- QTR_INS_BOT 底部 1/4 行插入指令（仅 BC7591）
- QTR_WT_TOP 顶部 1/4 行写入指令（仅 BC7591）
- WRITE_EXT 扩展位不译码显示
- QTR_INS_TOP 顶部 1/4 行插入指令（仅 BC7591）
- DECODE_EXT 扩展位译码显示
- SEG_OFF 段 OFF 指令
- COORD_OFF 坐标点 OFF 指令（仅 BC7591）
- SEG_ON 段 ON 指令
- COORD_ON 坐标点 ON 指令（仅 BC7591）
- BLINK_DIG_CTL 位闪烁控制指令
- GLOBAL_CTL 整体控制指令
- WRITE_ALL 全局写入指令
- BLINK_SPEED 闪烁速度控制指令
- DIM_CTL 亮度控制指令

- RESET_H 复位指令（作为 Cmd）
- RESET_L 复位指令（作为 Data）
- UART_SEND_0_H UART 发送 0 指令（作为 Cmd）
- UART_SEND_0_L UART 发送 0 指令（作为 Data）

clear() 清除显示和闪烁

函数使用格式：

```
clear();
```

该函数用来清除所有的显示和闪烁属性。

displayDec() 按10 进制显示数值

函数使用格式：

```
displayDec(Val, Pos, Width);
```

此函数将输入数值按十进制显示。只接受无符号数值，显示负值需用户自行显示 ‘-’号。如果高位数字超出了芯片的显示范围，超出部分将不会显示，也不会给出错误信息。

Val 为待显示数值，取值范围是 0~4,294,967,295。

Pos 是显示的位置。显示位置以最低位为准，Pos 值即最低位所在的数码管 DIG 序号，取值范围为 0-31。

Width 是显示宽度，有效数值范围为 0-255，在二进制表示中，低 7 位为显示宽度数值，最高位为是否显示前导 0 的控制位。即 Width 值为 1-127 时，如果设置显示宽度大于实际数字的宽度，超出部分将显示为空白，比如设置 Width 为 5，待显示的 Val 值为 132，则显示结果将为 “_ _ 132”（_ 代表空白），如果 Width 值为 133 (5|0x80，将 5 的二进制表示中最高位置为 ‘1’)，则显示结果将会是 “00132”。显示宽度有效范围为 1-32。当显示宽度设置小于实际数值的宽度时，超出部分将不予显示，而当显示宽度 Width 大于实际数值的宽度时，将根据 Width 的 bit7 决定超出部分显示空白或 ‘0’。如果数字的显示位置超出了实际芯片的显示范围，超出部分将被忽略。

setDisplayLowDigOnLeft() 和 setDisplayLowDigOnRight() 将影响数字显示的方向。当使用 setDisplayLowDigOnRight() 时，第 Pos 位显示最低位，而显示数值的较高位将依次显示在更高序号值的 DIG 位。而当使用 setDisplayLowDigOnLeft() 时，显示数值的较高位将依次显示在比 Pos 值更小的 DIG 位上。驱动库默认是 setDisplayLowDigOnRight() 的状态。

displayHex() 按16 进制显示数值

函数使用格式：

```
displayHex(Val, Pos, Width);
```

此函数将输入数值按 16 进制显示。此函数最大输入值为 0xffff，不过更大数字可以通过拆解和多次调用本函数完成显示。Pos 和 Width 的含义均与上面 displayDec() 中的含义相同，所不同的一点在于，对

于 16 进制显示, 当 Width 设置超过实际数值的宽度时, 超出部分将显示 0, 而不是十进制的空白。例如输入数值为 0xA5, 设置 Width 为 5, 则显示的结果将是 000A5

displayFloat() 显示浮点数

函数使用格式:

```
displayFloat (Val, Pos, Width, Frac);
```

此函数接受浮点数输入, 并显示出来。Pos 和 Width 的含义均与上面 displayDec() 中的含义相同, Frac 是小数部分的宽度, Width 是整个包括整数和小数部分的总宽度。此函数不包括小数点和符号位的处理, 用户需要单独使用段点亮指令显示小数点和可能需要的负号 “-”。

digitBlink() 位闪烁控制

函数使用格式:

```
digitBlink (Digit, OnOff);
```

按数码管位控制闪烁。此函数每次控制一个显示位的闪烁属性。第 16 位-第 31 位数码管的闪烁控制, 如果用户通过 sendCmd() 函数直接改变了状态, 再使用本函数对 16-31 位的闪烁进行控制时, 通过直接命令所做的设置可能会丢失。Digit 为待设置的显示位, 取值范围为 0-31; OnOff 为设置的状态, 1=闪烁, 0=不闪烁。

使用方法示例

下面的 Sketch 使用 Serial1 作为显示口，显示一个从 0-999 的计数，分别用十进制和十六进制显示，同时，还显示一个浮点数，从 1.23 开始，每次增加 0.01，每个显示会持续 1 秒，然后切换到另一个显示：

```
#include <UART_7Seg_Display.h>

#define LED_SERIAL Serial1 // 默认使用 Serial1 作为显示串口
// 如果您使用没有 Serial1 的型号，如 UNO，您需要去掉上面的语句，使用下面的软件串口设置
// #include <SoftwareSerial.h>
// #define LED_SERIAL swSerial
// SoftwareSerial swSerial(11, 12); // 创面软件串口实例，使用 IO 口 11(RX), 12(TX), (Rx 在本例中没有用到)

UART_7Seg_Display Disp(LED_SERIAL); // 建立显示驱动实例

uint16_t cnt=0;
float x=1.23;

void setup()
{
    Serial.begin(115200); // 初始化监听串口
    LED_SERIAL.begin(9600); // 初始化显示串口
    Disp.clear(); // 清除显示
}

void loop()
{
    Serial.print("Disaplying in decimal : ");
    Serial.println(cnt, DEC);
    Disp.displayDec(cnt, 0, 3); // 在 DIG-DIG2 按十进制显示 cnt
    delay(1000);
    Serial.print("Disaplying in hexadecimal : ");
```

```
Serial.println(cnt, HEX);
Disp.displayHex(cnt, 0, 4); // 在DIG-DIG3 按十六进制显示 cnt
delay(1000);
Serial.print("Disaplying in float : ");
Serial.println(x,2);
Disp.displayFloat(x, 0, 3, 2); // 在DIG-DIG2 显示浮点数 x, 小数点
后取 2 位
Disp.sendCmd(SEG_ON, 0x17); // 在DIG2 上显示小数点(段地址 0x17, 请参考芯片数据手册)
delay(1000);
Disp.sendCmd(SEG_OFF, 0x17); // 关闭小数点显示
cnt = cnt + 1;
if (cnt == 1000)
{
    cnt = 0;
}
x = x+0.01;
}
```