Journal of Wuhan Polytechnic

基于 U8G2 的 Arduino 图文程序剖析与仿真

彭　伟

（武汉城市职业学院　电子信息工程学院，湖北　武汉　430064）

摘　要：基于 Arduino 基本框架，对 U8G2 图形驱动库应用作深入剖析，包括两种页缓冲模式、全缓冲模式、无缓冲模式，剖析关键底层函数，在 Proteus 平台进行仿真应用测试，为微控制器资源受限时的图文程序开发提供参考。

关键词：Arduino；U8G2；Proteus 仿真；图文程序

中图分类号：TP368.2 / TP311.1　　**文献标识码：**A　　**文章编号：**1671-931X（2021）03-0112-04

DOI：10.19899/j.cnki.42-1669/Z.2021.02.021

Arduino 是全球最为流行的开源硬件之一，其简单的开发方式使开发者能够更关注创意与实现，大大节约成本缩减开发周期，大量开发者使用 Arduino 进入硬件、物联网等开发领域。Arduino 的 IDE 开发环境，可在 Windows、Macintosh OS X、Linux 上运行，针对大量应用涉及的液晶图文绘制问题，开发者迫切需要一种通用的液晶驱动库，其中的 U8glib 就是在 ARM、C51、Arduino 等 MCU 上都得到很好支持的图形绘制库，本文将就其最新版本 U8g2lib（以下简称 U8G2）在 Arduino 平台[1][4]上的显示器图文绘制开发进行剖析及仿真测试。

一、U8G2 应用剖析

U8g2 是单色 LCD、OLED 及 eInk 显示驱动库，所支持显示控制器包括 SSD13XX、T6963、UC16XX、ST75XX、KS0108 等，支持 I2C、SPI 及并行接口。U8g2 库包括所有的图形函数（包括绘制线条、方框、圆等），支持大量字体，需占用部分微控制器内存用于显示控制。使用 U8g2 显示驱动库须先掌握其构造函数，其构造器 5 个组成部分如表 1 所示。

表 1 中模式 1、2 均称为页缓冲（page buffer）模式，“F”模式称为全缓冲（full buffer）模式，此外还有 u8x8 模式（字符模式），仍以 KS0108 液晶为例，它有多种构造，如：U8G2_KS0108_128X64_1、U8G2_KS0108_128X64_2、U8G2_KS0108_128X64_F，它们的对应的 RAM 缓冲分别为 128 字节、256 字节、1024 字节 RAM，显然，其中前两种均为页缓冲（1 或 2 页），第三种为全缓冲。

（一）页缓冲模式 1、2 下的图文绘制程序

语句 U8G2_KS0108_128X64_1 u8g2(U8G2_R0, 8, 9, 10, 11, 4, 5, 6, 7,/*enable=*/ 18, /*dc=*/ 17, /*cs0=*/ 14, /*cs1=*/ 15,/*cs2=*/ U8X8_PIN_NONE, /* reset=*/ U8X8_PIN_NONE) 可构造对象 u8g2，它同时配置了显示屏的 D0~D7 引脚，另包括：E、DC（即 DI）、CS0、CS1 等引脚。

向显示屏写入命令、数据前，配置函数先执行：pinMode(16,OUTPUT) 与 digitalWrite(16,0)，它们将 RW 引脚拉低，然后执行 begin() 初始化 u8g2 对象，执行 u8g2.setFont(u8g2_font_6x10_tf) 设置字体。

由于 U8G2_KS0108_128X64_1 缓冲模式为 1，主循环须用 firstPage 及 nextPage 循环配合控制显示。改为 U8G2_KS0108_128X64_2 选择缓冲模式 2 时，

收稿日期：2021-01-27

作者简介：彭伟（1972- ），男，湖北武汉人，武汉城市职业学院电子信息工程学院教授，研究方向：算法设计、智能控制、应用系统开发。

其控制显示方法完全相同，页缓冲模式（选项为 1 或 2）下的 U8g2 典型显示框架如下：

```
u8g2.firstPage(); // 第一页（完成相关初始化）
do {
// 图文 1、图文 2、图文 3…. 系列绘制语句
// 其他绘制语句……
} while ( u8g2.nextPage() );    // 下一页循环
```

表 1　构造器组成部分表

序号	说明	示例
1	前缀	U8G2
2	显示控制器	UC1701、KS0108 等
3	显示器名称	DOGS102、128X64 等
4	缓冲大小（缓冲模式）	1、2 或 F（全缓冲 /Full）
5	接口	4W_SW_SPI 等

注：选项 F 只能在微控制器有足够 RAM 时采用，选项 1 和选项 2 适用于 RAM 较小的微控制器，对于可用 RAM 少于一个显示页的微控制器，可选择使用 u8x8 API。对于页（page），以 128x64 幅面 KS0108 为例，可将其理解为 1 个高 8 像素、宽 128 像素的显示“带区”，而不要误认为是 1 个整屏，1 个整屏被称为 1 帧；理解片块（tile）概念时，可将其理解为 8*8 像素的显示块。

以 U8G2_KS0108_128X64_1 为例：

缓冲模式 1：仅有 1 个显示页存储于微控制器 RAM，此时必须用 firstPage 与 nextPage 循环控制推送出所有页的显示（最后形成一个完整的显示）；

缓冲模式 2：与前者唯一差别是它每次缓存两页在微控制器 RAM，其显示速度比前者快 1 倍；

缓冲模式 F：整个显示帧缓冲全部保存于微控制器 RAM，clearBuffer 用于清除 RAM，sendBuffer 用于将微控制器 RAM 发送显示器显示。

KS0108 最多可显示 8 页（0～7 页），上述框架中，循环内的各绘制语句所绘制的图文内容，无论是从左到右，还是从上到下、或从下到上、或是其他情形跨页（或逆序）绘制显示，凡超出 0～7 页范围的部分均不会显示。无论待显示语句、显示内容有多少，显示信息密度如何，在 U8G2_KS0108_128X64_1 构造下，do while 总是共计循环 8 次，控制 8 页逐一输出（注：“_1”缓冲模式下微控制器所分配的页缓冲 RAM 空间仅为全屏像素空间的 1/8）。

do while 第 1 次循环，其内部所有绘制语句均被执行一遍，但只那些绘制语句所绘制内容恰好处于显示器第 0 页范围内，这些内容才会写入微控制器页 RAM 缓冲，然后送显示器显示，凡第 0 页范围之外的像素将被暂时被忽略。

do while 第 2 次循环，其内部仍是那一批绘制语句，同样将被再次“重复”执行一遍，但只那些绘制语句所绘制内容恰好处于显示器第 1 页范围内，相关内容才会写入 RAM 缓冲然后送显示器显示，凡第 1 页之外的像素同样将被暂时忽略。

do while 所有后续循环，其处理模式与之类似。

显然，每次循环时，循环内所有绘制语句都将被“重新”执行一遍，但仅属当前页范围内的显示内容（像素数据）才会被放入微控制器所分配的页缓冲 RAM，然后被输出到显示屏对应页位置显示，不在当前页范围内的内容本次将被暂时忽略。

将 U8G2_KS0108_128X64_1 后缀“_1”改为“_2”时，它相当于由单页缓冲模式切换到了双页缓冲模式，此时微控制器页缓冲 RAM 空间将定义为全屏像素空间的 2/8，也就是 1/4，因此上述 do while 循环将仅执行 4 次，因为第 1 次循环时第 0、1 共两页内容被写入微控制器页缓冲 RAM，第 2 次循环时第 2、3 共两页内容被写入页缓冲 RAM，依此类推。

（二）全缓冲模式下的图文绘制程序

将 U8G2_KS0108_128X64_1 后缀“_1”改为“_F”，即表示进入全缓冲模式，其显示控制参考代码如下：

```
u8g2.clearBuffer(); // 清显示缓冲
// 图文 1、图文 2、图文 3 等…. 系列绘制语句
u8g2.sendBuffer(); // 发送缓冲（送显示器显示）
```

该模式需微控制器分配足够的 RAM 空间（与显示器屏幕像素所占用空间相同），当然其显示速度相对而言也是最快的，所有显示语句不再需要被重复循环执行 n 次，所有绘制语句将一次性完成对显示 RAM 缓冲的写入，然后输出刷新显示。

（三）u8x8 无缓冲字符模式下的显示程序设计

u8x8 模式下 setup 函数须引入以下语句：

```
u8x8.begin();                    // 初始化 U8g2 对象
u8x8.setPowerSave(0);            // 非省电模式
```

主循环 loop 中须引入以下语句：

```
u8x8.setFont(u8x8_font_chroma48medium8_r);
u8x8.drawString(0,0,” Hello World!”); // 显示串
u8x8.refreshDisplay();           // 刷新
```

u8x8 模式代码要简单很多，大部分功能被忽略。

二、firstPage 与 nextPage 的内部实现

U8g2 的“_1”“_2”缓冲模式，适于在 RAM 有限的情况下使用，它们可大幅减少 RAM 分配，但这是以适当降低显示速度为代价的，其后提供关键支持的是 U8g2 的两个关键函数 firstPage 与 nextPage，它们分别调用的是 u8g2_FirstPage 与 u8g2_NextPage，这两个关键函数均由 u8g2 专门管理缓冲的 clib/u8g2_buffer.c 程序文件提供：

```
// 首页函数（完成相关初始化）
void u8g2_FirstPage(u8g2_t *u8g2) {
if ( u8g2->is_auto_page_clear ) {
u8g2_ClearBuffer(u8g2); } // 清缓冲
```

```
// 设置第 0 号片块行(即第 0 页)缓冲
u8g2_SetBufferCurrTileRow(u8g2, 0);
}
// 控制下页显示函数(KS0108:0~7 页)
uint8_t u8g2_NextPage(u8g2_t *u8g2) {
uint8_t row; // 行变量(实际为页索引变量)
// 发送 RAM 缓冲到显示器显示,首次为第 0 页
u8g2_send_buffer(u8g2);
row = u8g2->tile_curr_row; // 当前片块行(页)号
row += u8g2->tile_buf_height; // 递增
// 判断纵向是否越界
if ( row >= u8g2_GetU8x8(u8g2)->display_info->tile_height ) { //row ≥ tile 高度(相当于页高)
u8x8_RefreshDisplay( u8g2_GetU8x8(u8g2) );
return 0; // 返回 0(导致 do while 终止)
}
if ( u8g2->is_auto_page_clear ) {// 清缓冲
u8g2_ClearBuffer(u8g2); }
// 更新当前片块行(页)
u8g2_SetBufferCurrTileRow(u8g2, row);
return 1; // 返回 1(do while 继续)
}
```

do while 及 U8G2 相关关键函数执行流程如图 1 所示,最先执行的是 u8g2.firstPage(),即 u8g2_FirstPage,它针对第 0 页完成 RAM 缓冲初始化,u8g2_SetBufferCurrTileRow 是其调用的更底层函数,其内根据旋转模式等设置绘制回调函数,同时设置 row=0(行 =0,实际即页为 0)。

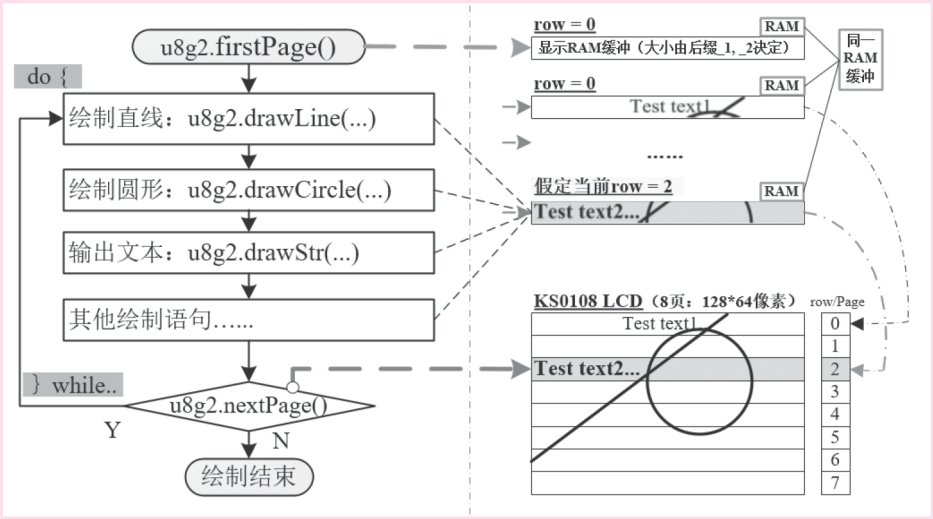

图 1　关键函数执行流程图

do while 之间是所有的绘制语句,以循环到 row=2 为例(指向第 2 行,即第 2 页),所有绘制语句所输出像素数据中,仅属于 row=2 范围内的像素数据被写入 RAM 缓冲,当执行 do while 后面的 u8g2.nextPage(),即 u8g2_NextPage 时,其内部 u8g2_send_buffer(u8g2) 将当前缓冲 RAM 中存放的第 2 页画面的像素数据发送显示器显示,其内同时调用 u8g2_SetBufferCurrTileRow 完成下一页,即第 3 页缓冲初始准备(row 被置为 3),循环再次进入执行 do while 时,循环中包括的所有 draw 语句再次全部重新执行,但仅有所绘制像素属于第 3 页的像素被写入 RAM 缓冲,其后操作依此类推,以当前“_1”模式为例,绘制程序在整个过程始中终使用同一大小的 RAM 缓冲,且其大小始终为全屏像素空间的 1/8。

三、仿真应用

仿真测试 U8g2lib 程序时,可选择用 Proteus[2] 绘制 Arduino+KS0108 电路,并基于 U8g2lib 编程,setup 函数将其配置为页缓冲模式代码前面已讨论过,此略。对 loop 主循环,其主要代码如下:

```
void loop() {
u8g2.firstPage();        // 第 1 页开始(初始配置)
do {
u8g2.drawStr(10,10," U8g2 display test!" );
u8g2.drawRFrame(40,0,80,63,5);        // 绘制圆角矩形
u8g2.drawCircle(80,25,20);            // 画圆 1
u8g2.drawCircle(10,10,5);             // 画圆 2
u8g2.drawCircle(30,30,6);             // 画圆 3
u8g2.drawCircle(40,40,20);            // 画圆 4
// 以下显示内容用于跟踪相关参数
// 当前片块行(页号)0 ~ 7
Serial.println(u8g2.getPageCurrTileRow());
// 当前像素行(递增 8)
Serial.println(u8g2.getU8g2()->pixel_curr_row);
delay(1000);            //延时以便观察逐页缓冲显示效果
```

```
} while ( u8g2.nextPage() );    // 下一页循环
// 以下输出用于跟踪输出 U8g2lib 相关参数
// 片块高(页高)
Serial.println(u8g2.getBufferTileHeight());
// 像素缓冲宽度(显示为 128)
Serial.println(u8g2.getU8g2()->pixel_buf_width);
// 像素缓冲高度(显示为 8)
Serial.println(u8g2.getU8g2()->pixel_buf_height);
while (1);    // 需反复刷新观测显示时可删除此行
}
```

程序仿真[3]测试运行效果如图 2。

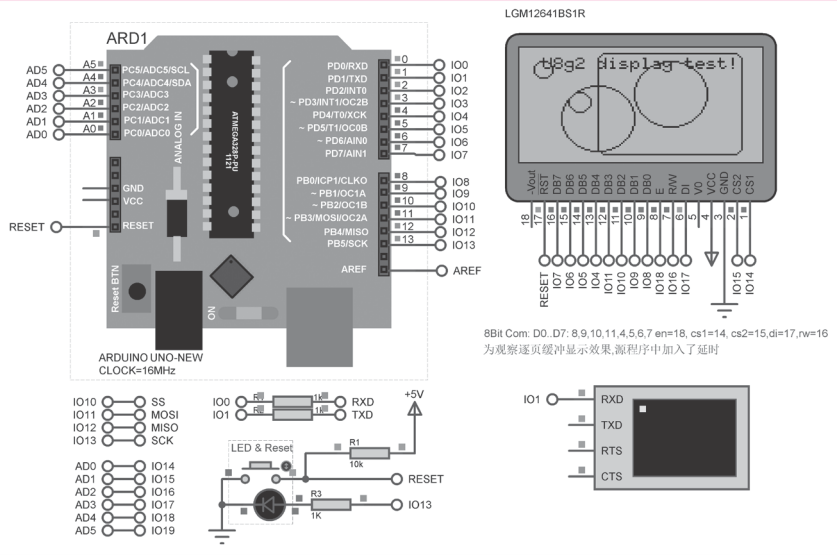

图 2　程序仿真测试运行效果图

四、总结

U8g2lib 为 Arduino 平台图文绘制程序设计[5]提供了极大便利,研究掌握其内部具体实现机理与细节,不仅可使开发者更好的掌握其应用方法与技巧,同时为微控制器系统资源受限时的嵌入式程序开发提供了重要参考策略。

参考文献:

[1] 王红敏,王燕,刘军强,等.基于Arduino控制的OLED显示模块的电子实践教学研究[J].高教学刊,2021,(3):7-11.

[2] 田社平,方向忠,张峰.VSPD和Proteus串口通信教学实验[J].实验室研究与探索,2018,(9):211-214.

[3] 杨丽新,马迎.基于Proteus仿真的教学机器人智能小车设计[J].实验室研究与探索,2018,(8):139-143.

[4] 孙建振,王振.基于ARDUINO的多功能智能LED点阵屏[J].电脑知识与技术,2020,(8):220-221.

[5] 胡天林,李继芳.基于Arduino的移动机器人实训平台设计[J].实验技术与管理,2020,(12):108-111.

[责任编辑: 胡大威]

Analyzing and Simulation of Arduino Graphics Based on U8G2

PENG Wei

(School of Electronic Information Engineering, Wuhan Polytechnic, Wuhan430074, China)

Abstract: Based on the basic framework of Arduino, this paper analyzes the application of the graphic driver library of u8g2, including two kinds of page buffering mode, full buffering mode and no buffering mode, analyzes the key underlying functions, and carries out the simulation application test on the Proteus platform, which provides a reference for the graphic program development when the microcontroller resource is limited.

Key words: Arduino; u8g2; proteus simulation; graphic