# RGB Keyboard V2.1版本

## 使用的库：

1. HID-Project库：键盘USB功能主要支撑以及扩展

<https://github.com/NicoHood/HID>

1. Keypad库：为矩阵键盘识别提供一些支持
2. Adafruit\_NeoPixel库：驱动WS2811芯片驱动RGB LED，并提供一部分灯效
3. EEPROM库：官方库，为用户设置存储提供支持
4. Wire库：IIC通信库，未直接使用，仅为OLED显示库提供支持
5. SSD1306Ascii & SSD1306AsciiWire库：OLED显示驱动库，可显示字符串
6. SchedulerARMAVR库：Scoop库中为AVR单片机设计的多线程任务调度器库

## 文件列表

|  |  |
| --- | --- |
| RGBkeyboard.ino | 程序入口 |
| KeyScan.cpp | 按键扫描功能，提供按键扫描所需功能 |
| Pressed.cpp | 提供按键下降沿事件处理 |
| Released.cpp | 提供按键上升沿事件处理 |
| HOLD.cpp | 宏进程 |
| LED.cpp | LED驱动及灯效 |
| OLED.cpp | OLED驱动及基本状态显示函数 |
| Macro.cpp | 用于检测宏状态 |
| Keydefine.cpp | 定义layer1和layer2层按键 |

## 特性

V2.1

1. 修正了抬起其他按键使得当前按键宏失效的问题
2. 修正了同时只能识别6个按键的问题
3. **因2的修改，下文中所有函数名中Keyboard前需要添加NKRO，即Keyboard.xxx()改为NKROKeyboard.xxx()**
4. 添加了多媒体控制器功能，详情见最后多媒体键定义章节

**V2.0**

1. 修改按键宏使能方案，17号按键在layer1&2层用作FN多功能组合键
2. Layer1层按住FN键将使得下一次按下被引导进宏，宏也可以发送一个普通按键或者字符串
3. Layer2层宏定义不再需要将按键定义为KEY\_RESERVED，只需短按FN键即可起用
4. 为FN按键状态添加OLED显示

V1.3

1. 添加了显示函数，可以通过简单修改一个变量值来控制最后一行显示的内容
2. 添加了新的显示更新方式，当显示控制变量改变时刷新屏幕

V1.2

1. 修正LED灯效切换时延时的BUG
2. 添加LED亮度多级调节
3. 更换了灯效一和二
4. 优化了用户灯光设置存储
5. 小幅优化代码效率

V 1.1

删除了无效的代码，精简逻辑减少资源消耗

V 1.0：

1. 使用Scoop库设计四个合作多线程
2. 可以定义死循环宏而不影响其他任务
3. Layer1和layer2层按键使用数组设置方便快捷
4. LED专用设置层，定义4种亮度，4种灯效，还有12个按键空余
5. OLED目前可简单显示键盘状态
6. Layer3专用宏定义层，可定义死循环宏
7. layer2层可定义按住循环宏或单次宏
8. layer1层仅可定义按键，为开机默认层
9. 用户可以设置LED灯效，并且该灯效掉电可保存，具有至少100，000次更改寿命

## 变量定义

按键号定义：

按键号1~20，自左到右，自上到下，Z字型排列。

层变量定义：

key\_state 值为0~2，使用mode按键自加，由RGBkeyboard.h文件中MAX\_KEY\_LAYER宏定义限定最大值为2。

宏使能标志定义：

macro\_flag变量定义了宏使能标志，此值用于在键盘扫描线程和宏专用线程中传递某个按键是否应当启用宏。取值0~20，值0禁用宏，1~20使能对应按键的宏。因此同时只能激活一个宏！！

LED号定义：

LED号0~19，与按键号同样排序

LED相关变量：

led\_state取值为0或1，指示键盘模式（0）或者灯效设置模式（1）。

led\_layer取值为int，目前定义为0，1，2，3，分别代表四种灯效

## Layer1 & layer2层按键定义方式

若要定义layer1和layer2层的按键，只需打开keydefine.cpp，对照 键值表.txt 文件更改数组即可。

若需要在某键位定义一个宏，需要将该键位定义为KEY\_RESERVED来使能该键位的宏，宏会被引导进HOLD.cpp 文件中的宏处理函数，并交由另一个专用线程来处理

## Layer1&layer2层宏定义方式

~~若要定义按键宏，需要先定义此按键为KEY\_RESERVED！！~~（弃用）

FN键用于引导进入宏，层1FN按下有效，进入宏后放开FN宏仍然可以运行，但松开按键会失能宏。层2FN切换有效性，进入宏后失能FN后宏仍然运行，直到松开

宏在HOLD.cpp文件中定义，每个按键均有一个对应的宏处理函数。因此函数与layer3宏专用层复用，需要使用

if(key\_state == 层号-1)

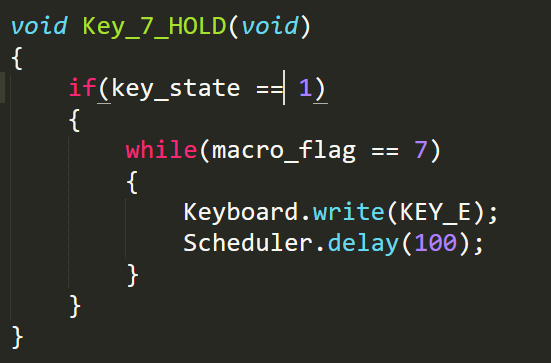
函数来限定宏对应的按键层。

Layer2层宏均可直接定义死循环，因此若想要令宏按下时循环触发，使用

while(macro\_flag == 按键号)

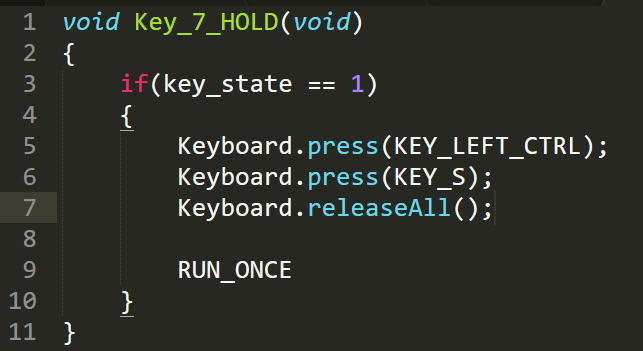
来设置循环触发的宏。

例：按住按键7，以100ms间隔重复发送小写字母E



Layer2层也可以定义单次运行宏，不使用while循环并在宏最后定义RUN\_ONCE即可

例：按下按键7，保存文档

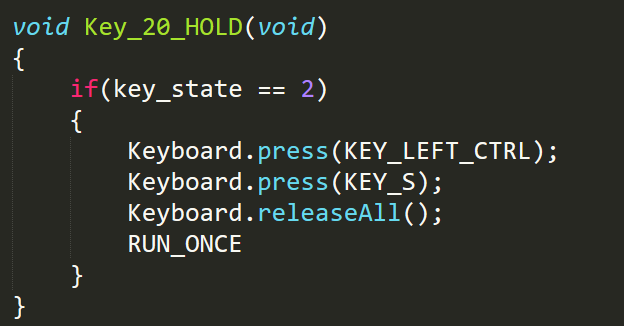


注意！延时必须使用Scheduler.delay(time)函数，否则会导致多线程失效！！

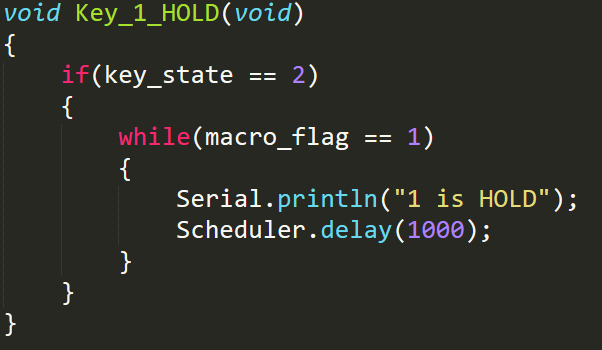
## Layer3层定义宏方式

Layer3层与layer2层不同，此层默认使用按键切换宏是否激活，第一次按下按键激活宏，第二次按下按键关闭宏。此层也具备两种定义方式，可以定义一个死循环宏来彻底解放双手，也可以使用宏定义RUN\_ONCE来使宏只在按键按下后触发一次。

例1：按下按键保存文档，一次触发



例2：按下按键重复发送串口消息1 is HOLD直到再次按下按键结束

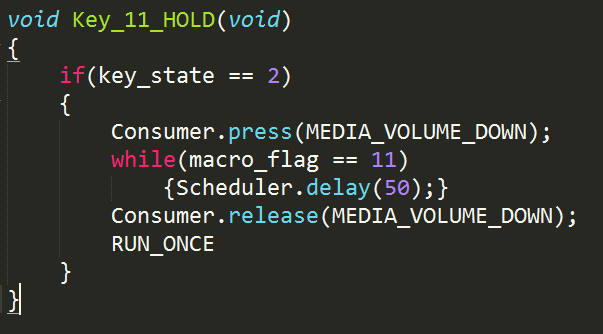


## 如何在HOLD函数中实现普通按键功能

普通按键功能主要使用press（）函数和release（）函数来实现，因此可以在HOLD

数中使用press和release函数来实现普通按键

例如：



使用while(macro\_flag == 按键号)来等待按键抬起再继续执行，因同时只能有一个宏在运行，这种方法是安全的。

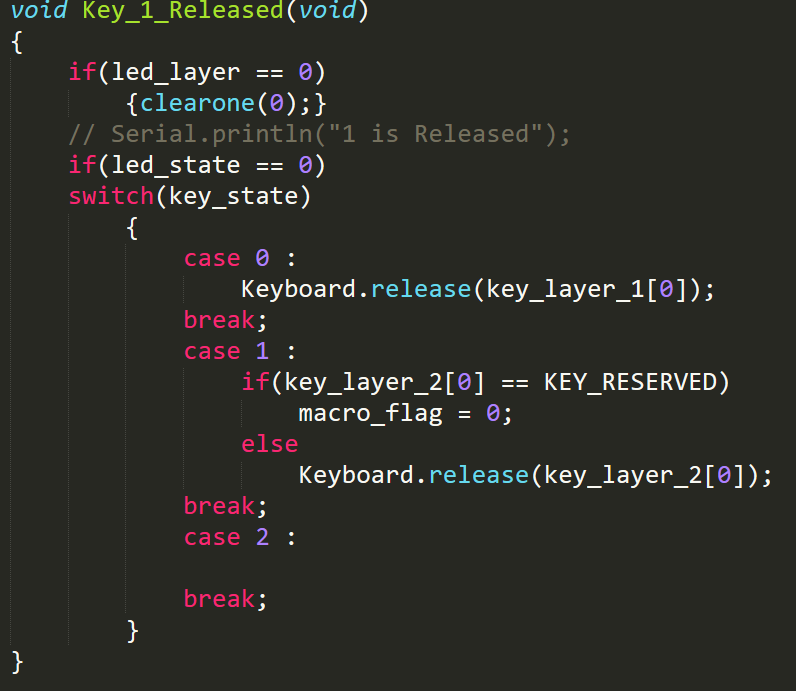
注意：while中必须使用延时函数，否则会卡死程序。

## ~~进阶宏定义~~（本节已失效）

~~本程序可以进行进阶的宏定义，只需在Pressed.cpp和Released.cpp文件中定义如何处理宏使能标志即可。默认的宏使能标志为marco\_flag变量进行如下处理：~~

~~~~

~~这是一个完整的按键按下事件处理函数，其中最开始处理的是按键灯效一。随后if语句来区别这个按键是用作键盘还是用作灯光设置，随后按照相应的按键层来处理按键事件。可以看到在按键层二按键按下时将按键一宏使能了，而在层三则将按键宏反相了。这里可以详细的定义按键宏变量的改变方式。对应层二，在Released.cpp文件中有：~~

~~~~

~~将按键宏失能的语句。~~

~~若要启用其他按键的宏，只需将marco\_flag赋值为对应的按键号即可~~

~~例：使能按键二十的宏~~

~~marco\_flag = 20;~~

~~失能按键二十的宏~~

~~marco\_flag = 0;~~

## 宏编程时可能会用到的函数

|  |  |
| --- | --- |
| Scheduler.delay(time) | 延时time毫秒，必须使用此延时函数 |
| NKROKeyboard.press(key) | 按下按键key（必须与release函数成对出现） |
| NKROKeyboard.release(key) | 放开按键key |
| NKROKeyboard.write(key) | 发送按键key |
| NKROKeyboard.print(string) | 发送一个字符串string |
| NKROKeyboard.println(string) | 发送一个字符串string并回车 |
| NKROKeyboard.releaseAll(void) | 释放所有按键 |
| RUN\_ONCE | 宏定义，令宏单次运行，无需分号 |

## 程序对按键事件的处理

Pressed.cpp逻辑过程

开始

按键按下

判断按键层

FN有效？

FN有效？

层1

层2

层1按键

层2按键

层1宏

层2宏

层3宏

层3

是

是

结束

Released.cpp文件负责释放对应按键，清空相应宏

## 编写灯效

灯效写在LED.cpp文件中，提供了三种基本灯效：脉冲填充，同步彩虹渐变，彩虹连续脉冲，以及一个彩虹数据运算支持函数。

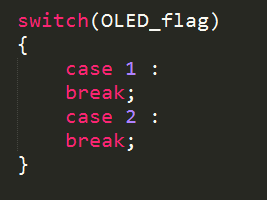
在编写灯效时应当注意，因灯效多数需要依靠for循环来实现，但是for中使用延时就可能使得LED切换灯效时不能及时反应，故循环中必须使用宏定义IF\_LED\_LAYER\_CHANGE来使得切换灯效时可以及时反应。

## 编写灯效时可能会用到的函数

|  |  |
| --- | --- |
| Scheduler.delay(time) | 延时time毫秒，必须使用此延时函数 |
| strip.setPixelColor(n,color) | 设置n号LED为color（32位）色 |
| strip.Color(R, G, B) | 输入RGB色值，返回32位颜色编码 |
| strip.show(void) | 将设置更新到LED上 |
| strip.numPixels(void) | 返回LED数量 |

## OLED显示

Oled显示函数OLED\_Display定义于OLED.cpp



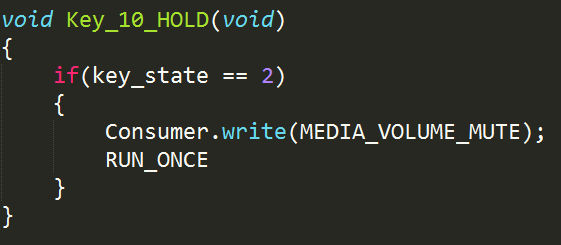
部分用于定义OLED\_flag改变时屏幕应当显示的内容，标志为0时最后一行不显示任何内容，其他值由switch函数选择显示

## 多媒体键定义

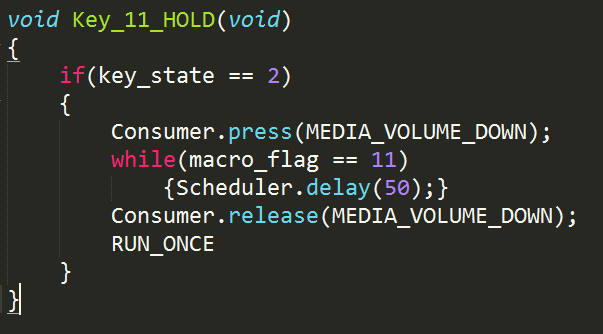
若想定义多媒体键，需要使用宏定义的方式来定义在文件HOLD.cpp中，因多媒体键与普通按键数据结构不同，因此无法使用普通按键宏定义的方式来定义多媒体键。

定义多媒体键使用Consumer.write(键值)等函数来定义，详细函数见下节表。键值存放于多媒体键值表.txt中

例1：层3-10号按键定义为静音键



例2：层3-11号按键定义为音量减少，长按有效的写法



## 定义多媒体按键时可能会用到的函数

|  |  |
| --- | --- |
| Scheduler.delay(time) | 延时time毫秒，必须使用此延时函数 |
| Consumer.press(key) | 按下按键key（必须与release函数成对出现） |
| Consumer.release(key) | 放开按键key |
| Consumer.write(key) | 发送按键key |
| Consumer.releaseAll(void) | 释放所有按键 |
| RUN\_ONCE | 宏定义，令宏单次运行，无需分号 |