09

第九章 自定义表情板

DFROBOT
DRIVE THE FUTURE

前情回顾

在上个章节里，我们学习了RGB表情板内置的23种表情和怎么使用myMax.show-Face、myMax.backward函数。

本章内容

自己定义表情的图案和颜色。

本章知识点

1. 学习表情板自定义的方法；
2. 学习数组的用法；
3. #define 定义方法；
4. myMax.customFace函数的使用；

一、编写并下载程序

打开 ArduinoIDE，将下面的代码输入到编辑区中

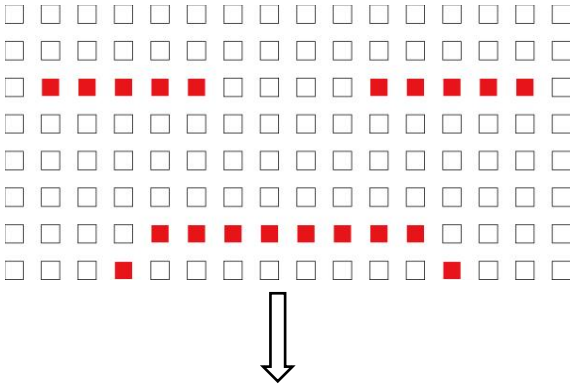
```
/*
程序功能：显示自定义表情。
作　　者：DFRobot
*/
#include <DFRobot_MAX.h>     //载入 MAX 驱动库
DFRobot_MAX myMax;           //载入 MAX 驱动函数
#define RED          1       //数字 1 代表的是红色
#define GREEN        2       //绿色
#define YELLOW       3       //黄色
#define BLUE         4       //蓝色
#define PURPLE       5       //紫色
#define CYAN         6       //蓝绿色
#define WHITE        7       //白色
static int8_t Emoticons[128] = {
      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, //0
      0,1,1,1,1,1,0,0,0,0,1,1,1,1,1,0, //1
      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, //2
      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, //3
      0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0, //4
      0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0, //5
      0,0,0,0,1,0,1,0,0,1,0,1,0,0,0,0, //6
      0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0, //7
   };
void setup() {
   myMax.begin();              //初始化 MAX
   myMax.clearScreen();        //表情板清屏
}
void loop() {
    myMax.customFace(Emoticons, PURPLE);   delay(2000);   //显示紫色的自定义表情；延时 2s。
    myMax.clearScreen();   delay(1000);          //清屏后延时 1s
}
```

上传完毕后拨开 MAX 的开关，我们就能看见自己自定义的表情了。

二、自定义表情板

如何自定义表情板

MAX 自定义表情板是利用数组，简单的点亮你所需要的位置（将想点亮的位置置高电平，如下图）。



```
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,    //0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,    //1
0,1,1,1,1,0,0,0,0,1,1,1,1,1,0,      //2
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,    //3
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,    //4
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,    //5
0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,    //6
0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,      //7
```

什么是数组

数组是指很多同样的数据类型在内存中连续排列的形式，作为数组元素的各个数据会通过连续的编号被区分开来。这个编号被称为索引。

例如：int n[8]={0,0,1,0,0,0,0,1};

我们声明一个 int 型的整型变量，取名为 n。然后用 8 个数值（也是 8 个元素）来初始化这个数组，但如何获取数组中的元素呢，这就需要索引号。例程中的数组从 0 开始索引，索引号为 0~7,在这里元素 3 对应的索引号为 2（n[2]）,数值为 1。

一般我们常用的数组类型为一维数组和二维数组。一维数组是最简单的数组，使用一维数组只需要经过定义、初始化和应用等。上面的例程就是一维数组。

二维数组在概念上是二维的，就是说有两个方向上变化，二维数组被称为“数组的数组”。

类型说明符：

数组名[常量表达式][常量表达式]。

例：int a[3][4]={

{0,1,1,0},

{1,0,0,1},

{1,0,1,1}

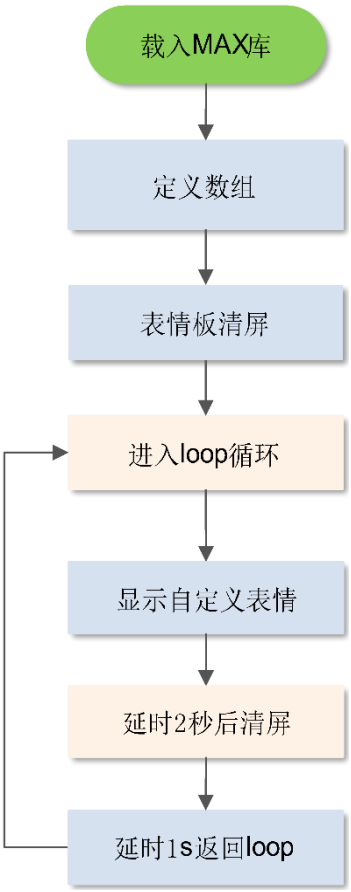
};

//定义 a 为 3*4（3 行 4 列）的数组；其中 a 有 3 个元素，分别为 a[0]，a[1]，a[2]，每个元素都是一个一维数组，各包含 4 个元素。

在本章节中，我们用一维数组比较方便。

三．代码回顾

与 MAX 内置的表情相比，自定义表情板可以自己设计图案，而且在程序设计方面也比较简单，直接利用数组定义好我们所需要的表情，然后在 loop 循环中调用出来就可以了。则程序的运行流程如右图:



代码回顾与分析

首先，定义 7 个常量，代表 7 种颜色。

```
#define RED       1
#define GREEN     2
......
#define CYAN      6
#define WHITE     7
```

在上一章我们知道了数值 1~7 分别代表的是 7 种不同的颜色，但偶尔我们也会忘记其中的某一个数值代表的是什么颜色，所以这章我们学习一个新的知识点——“无参宏定义”来更明确的表示这 7 个数值代表的是那些颜色。

“define”为无参宏定义，其含义是宏名后面不带参数。其定义的一般形式为：

#define 标识符 字符串 ,其中的“#”表示这是一条预处理命令。凡是以“#”开头的均为预处理命令。例如：“#define RED 1”它的意思就是：“给 1 取一个名字叫做 RED”,这样我们就明白：1 就是 RED 这个颜色了。

使用宏定义的优点:

方便程序的修改；使用宏定义可以用宏代替一个程序中经常使用的常量，这样要修改常量，不用在程序中一个一个的修改，只需要修改宏定义。

提高了程序的运行效率；使用带参数的宏定义可完成函数调用功能，这样减少了系统的运行，提高了运行效率；在使用起来让程序更加模块化，便于组织；还可以重复利用。

然后再定义表情板数组：

```
static int8_t Emoticons[128] = {
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, //0
  0,1,1,1,1,1,0,0,0,0,1,1,1,1,1,0, //1
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, //2
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, //3
  0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0, //4
  0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0, //5
  0,0,0,0,1,0,1,0,0,1,0,1,0,0,0,0,  //6
  0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0, //7
```

一维数组

元素

表情板是一个 16*8 的点阵屏，上面一共有 128 颗 LED 灯，所以我们建立了一个含有 128 个元素的一维数组。索引号由第一个元素 0 开始到第 127 个元素截止。

因为在像素点地址的寄存器里，高四位代表 Y 坐标(0<=Y<8)，低四位代表 X 坐标(0<=X<16)，所以我们定义一个 8 位数的静态全局变量：

```
static int8_t Emoticons[128];
```

静态全局变量在声明它的整个文件都是可见的，而在文件之外是不可见的。

如果将 static int8_t Emoticons[128]; //定义静态全局变量

改为

```
int8_t Emoticons[128];    //定义全局变量
```

程序照样正常运行；全局变量虽然可以实现变量在文件中的共享，但如果其他文件定义了相同名字的变量，容易发生冲突。而静态全局变量不会被其他文件所用，也不会与其他同名字的变量发生冲突。

定义好表情后，先利用函数 myMax.clearScreen();将表情板清屏，然后在 loop 循环中用函数 myMax.customFace()让自定义的表情在表情板上显示出来。

函数格式：

```
myMax.customFace(定义的数组名，颜色序号 1~7);
```

代码例程：

```
myMax.customFace(Emoticons, 5); //用紫色显示自定义表情
```

这样就完成了表情的自定义。后面想要更改成其他的图案，只需要将要显示图案的位置——置高电平（置 1）就行了。