引脚分配：

marlin原生固件中，对于引脚的需求根据打印机功能分为：

1 电机控制，

2 限位开关，

3 加热头及热床，

4 风扇控制，

5 传感器数据获取，

6 状态LED。

（当然这些只是基本功能，还有SD卡，LCD等拓展引脚）

1电机控制

电机控制又分为四个部分：

1.1 X轴电机

1.2 Y轴电机

1.3 Z轴电机

1.4 挤出机电机

2限位开关

限位开关具体分为三个部分：

2.1 X轴限位开关

2.2 Y轴限位开关

2.3 Z轴限位开关

3加热头及热床

加热头及热床具体分为两个部分：

3.1 加热头

3.2 热床

4风扇控制

5传感器数据获取

传感器数据获取具体分为两个部分

5.1 加热头传感器数据获取

5.2 热床传感器数据获取

6 状态LED

在arduino中对于引脚的定义有几条与之相关的命令：

#define X\_STEP\_PIN 54

pinMode(X\_STEP\_PIN,OUTPUT);

pinMode(X\_STEP\_PIN,INPUT);

WRITE(PHOTOGRAPH\_PIN, LOW);

WRITE(PHOTOGRAPH\_PIN, HIGH);

代码也很简单，顾名思义：

第一行是宏

第二、三行是设置输入或者输出

第四、五行是设置高低电平

相比较之下，STM32的引脚定义就复杂些了。

举例：

GPIO\_InitTypeDef GPIO\_InitStructure;

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_9; //TX

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP;

GPIO\_Init(GPIOA, &GPIO\_InitStructure);

代码解释：

第一行：定义结构体

第二行：设置pin值

第三行：设置输出速度

第四行：设置I/O口的模式

第五行：将之前的值赋值给结构体

因为STM32是面向有编程基础的程序员的，所以他的I/O口定义将I/O口抽象成为了一个结构体。定义时就是给结构体中的内容赋值。

在说明了两种编程的方法之后，现在要考虑如何将程序从arduino转化为STM32。

原则：

1尽可能少修改现有代码

解决方法：

在pin.h中的宏定义里面只修改引脚的名字，对于具体的I/O口定义再添加一个pin.c，然后在里面添加具体的针对每个引脚的定义。

具体引脚分配：

1 XYZE四个步进电机，每个电机3个I/O口，一共需要12个引脚，都为数字输出脚

2 XYZ三个轴上的限位开关，每个轴上一个，一共需要3个，都为数字输入脚

3 加热头和热床，一共需要2个，都为数字输出脚

4 风扇控制，一共需要1个，都为数字输出脚

5 温度传感器，一共需要2个，都为模拟输入脚

6 状态LED，一共需要1个，为数字输出脚

综上：一共需要21个引脚。Stm32f103c8t6完全可以提供这些引脚。

引脚分配如下表：

|  |  |
| --- | --- |
| 引脚名称 | 具体引脚 |
| X\_EN | PA3 |
| **X\_STEP** | **PA4** |
| **X\_DIR** | **PA5** |
| Y\_EN | PA6 |
| **Y\_STEP** | **PA7** |
| **Y\_DIR** | **PB0** |
| Z\_EN | PB1 |
| **Z\_STEP** | **PB5** |
| **Z\_DIR** | **PB6** |
| E\_EN | PA8 |
| E\_STEP | PA11 |
| E\_DIR | PA12 |
| X\_ENDSTOP\_MIN | PB10 |
| Y\_ENDSTOP\_MIN | PB11 |
| Z\_ENDSTOP\_MIN | PB12 |
| HEAT\_HOTEND | PB8 |
| HEAT\_BED | PB9 |
| FAN | PB13 |
| TEMP\_HOTEND | PA1 analog input |
| TEMP\_BED | PA2 analog input |
| STATUS\_LED | PB14 |