Marlin 固件基本配置

概述

众所周知，Sprinter 固件是之前用的比较多的 3D 打印机固件，而 Marlin 固件和 Repetier-firmware 固件都是由其派生而来。而且这两款固件的用户群非常活跃，而 Sprinter 固件已经没有人维护了。在这二者中，Marlin 固件的使用更加广泛，很多打印机控制软件都兼容 Marlin 固件。一般用户在使用 Marlin 固件的时候只需要改变一下 Configuration.h 文件中的一些参数即可非常方便，这对一般 3D 打印玩家来说可是非常好的福利哟。这份指南是一份简单的用户指南，告诉用户设置的基本信息、怎么运用这些设置、根据不同的需求制定特色功能。

Marlin 固件 GitHub 地址：https://github.com/ErikZalm/Marlin

Marlin 固件特点

Marlin 相对于 Sprinter 有很多优点，具体为以下几点：

1. 预加速功能（Look-ahead）：
   Sprinter 在每个角处必须使打印机先停下然后再加速继续运行，而预加速只会减速或加速到某一个速度值，从而速度的矢量变化不会超过 xy_jerk_velocity。要达到这样的效果，必须预先处理下一步的运动。这样一来加快了打印速度，而且在拐角处减少耗材的堆积，曲线打印更加平滑。
2. 支持圆弧（Arc Support）
   Marlin 固件可以自动调整分辨率去以接近恒定速度打印一段圆弧，得到最平滑的弧线。这样做的另一个优点是减少串口通信量。因为通过 1 条 G2/G3 指令即可打印圆弧，而不用通过多条 G1 指令。
3. 温度多重采样（Temperature Oversampling）
   为了降低噪声的干扰，使 PID 温度控制更加有效，Marlin 采样 16 次取平均值去计算温度。
4. 自动调节温度（AutoTemp）
   当打印任务要求挤出速度有较大的变化时，或者实时改变打印速度，那么打印速度也需要随之改变。通常情况下，较高的打印速度要求较高的温度，Marlin 可以使用 M109 S<mintemp> B<maxtemp> F<factor>指令去自动控制温度。
   使用不带 F 参数的 M109 指令不会自动调节温度。否则，Marlin 会计算缓存中所有移动指令中最大的挤出速度（单位是 steps/sec），即所谓的“maxerate”。然后目标温度值通过公式 T = tempmin + factor*maxerate，同时限制在最小温度（tempmin）和最大温度（tempmax）之间。如果目标温度小于最小温度，那么自动调节将不起作用。最理想的情况下，用户可以不用去控制温度，只需要在开始使用 M109 S B F，并在结束时使用 M109 S0。
5. 非易失存储器（EEPROM）
   Marlin 固件将一些常用的参数，比如加速度、最大速度、各轴运动单位等存储在 EEPROM 中，用户可以在校准打印机的时候调整这些参数，然后存储到 EEPROM 中，这些改变在

打印机重启之后生效而且永久保存。

6. 液晶显示器菜单（LCD Menu）
   如果硬件支持，用户可以构建一个脱机智能控制器（LCD 屏+SD 卡槽+编码器+按键）。用户可以通过液晶显示器菜单实时调整温度、加速度、速度、流量倍率，选择并打印 SD 卡中的 G-Code 文件，预加热，禁用步进电机和其他操作。比较常用的有 LCD2004 只能控制器和 LCD12864 只能控制器。
7. SD 卡内支持文件夹（SD card folders）
   Marlin 固件可以读取 SD 卡中子文件夹内的 G-Code 文件，不必是根目录下的文件。
8. SD 卡自动打印（SD card auto print）
   若 SD 卡根目录中有文件名为 auto[0-9].g 的文件时，打印机会在开机后自动开始打印该文件。
9. 限位开关触发记录（Endstop trigger reporting）
   如果打印机运行过程中碰到了限位开关，那么 Marlin 会将限位开关触发的位置发送到串口，并给出一个警告。这对于用户分析打印过程中遇到的问题是很有用的。
10. 编码规范（Coding paradigm）
    Marlin 固件采用模块化编程方式，让用户可以清晰地理解整个程序。这为以后将固件升级到 ARM 系统提供很大的方便
11. 基于中断的温度测量（Interrupt based temperature measurements）
    一路中断去处理 ADC 转换和检查温度变化，这样就减少了单片机资源的使用。
12. 支持多种机械结构
    普通的 XYZ 正交机械，CoreXY 机械，Delta 机械以及 SCARA 机械。

基本配置

使用 Arduino IDE 打开 marlin.ino，切换到 Configuration.h 即可查看并修改该文件。或者使用任何一款文本编辑器（notepad,notpad++等）直接打开 Configuration.h 也可以。Marlin 固件的配置主要包含一下几个方面：

1. 通讯波特率
2. 主板类型，所使用的主板类型
3. 温度传感器类型，包括挤出头温度传感器和加热床的温度传感器
4. 温度配置，包括喷头温度和加热床温度
5. PID 温控参数，包括喷头温度控制和加热床温度控制
6. 限位开关
7. 4 个轴步进电机方向
8. X/Y/Z 三个坐标轴的初始位置
9. 打印机运动范围
10. 自动调平
11. 运动速度
12. 各轴运动分辨率
13. 脱机控制器

根据笔者的经验来说，Marlin 固件中的 Configuration.h 将各个配置模块化，非常便于阅读及修改，而且注释非常详细，英文好的朋友可以很容易地理解各参数的意义。注意到 Marlin 固件使用 C 语言编写，“//”后面的是注释语句，不会影响代码的作用。另外 Marlin 固件中大

量使用#define，简单来讲，就是定义的意思，包括定义某个参数的数值，定义某个参数是否存在。

最开始的两行非注释语句是定义固件的版本和作者。缺省的版本号就是编译时间，这个可以不用修改，只需要把作者改为自己的名字即可，注意不能包含中文，不然会乱码。

```
#define STRING_VERSION_CONFIG_H __DATE__ " " __TIME__ // build date and time
#define STRING_CONFIG_H_AUTHOR "www.abaci3d.cn" // who made the changes.
```

电脑和打印机通过串口进行通讯，要定义好端口和波特率，在此定义的是 3D 打印主板的端口和波特率，端口号使用默认的 0 就可以了。Marlin 固件默认的波特率是 250000，也可以修改为其他值，比如 115200，这是标准的 ANSI 波特率值。

```
#define SERIAL_PORT 0
#define BAUDRATE 250000
```

下面定义主板类型，Marlin 固件支持非常多种类的 3D 打印机主板，比如常见的 RAMPS1.3/1.4、Melzi、Printrboard、Ultimainboard、Sanguinololu 等控制板。需要注意的是不同主板使用不同的脚口和数量，如果该定义和 Arduino IDE 中使用的主板不一致，肯定会导致编译不通过。笔者使用的是 RAMPS1.4 并且 D8、D9、D10 控制的是一个喷头加热、一个加热床加热和一个风扇输出，因此定义为 33。

```
#ifndef MOTHERBOARD
#define MOTHERBOARD 33
#endif
```

接下来是定义挤出头的个数及电源类型，笔者使用的是单喷头打印机，因此定义为 1。电源有两种类型可以选择，1 表示开关电源，2 表示 X-Box 360 203 伏电源，一般都使用的是开关电源，因此定义为 1。

```
#define EXTRUDERS 1
#define POWER_SUPPLY 1
```

接下来定义温度传感器类型，包括每个喷头使用的温度传感器（如果是多喷头）和加热床的温度传感器类型，常用的温度传感器有电热偶和热敏电阻两大类，热敏电阻又分为很多种。目前的 3D 打印机主要用的是热敏电阻，具体是哪种热敏电阻需要自己判断或询问卖家，不出意外的话，都是 100k ntc 热敏电阻，即 1。根据注释，1 要求 4.7k 的上拉电阻，而根据 RepRap wiki，几乎所有的 3D 打印机都使用了 4.7K 的热敏电阻上拉电阻。笔者观察了几种电路板的电路图，发现都使用了 4.7K 的上拉电阻，如图 1 所示。

```
// 1 is 100k thermistor - best choice for EPCOS 100k (4.7k pullup)
```

笔者的打印机为单喷头，因此第一个喷头的温度传感器配置为 1，其他配置为 0（0 表示没有使用），加热床的温度传感器也配置为 1。

```
#define TEMP_SENSOR_0 1
#define TEMP_SENSOR_1 0
#define TEMP_SENSOR_2 0
#define TEMP_SENSOR_BED 1
```

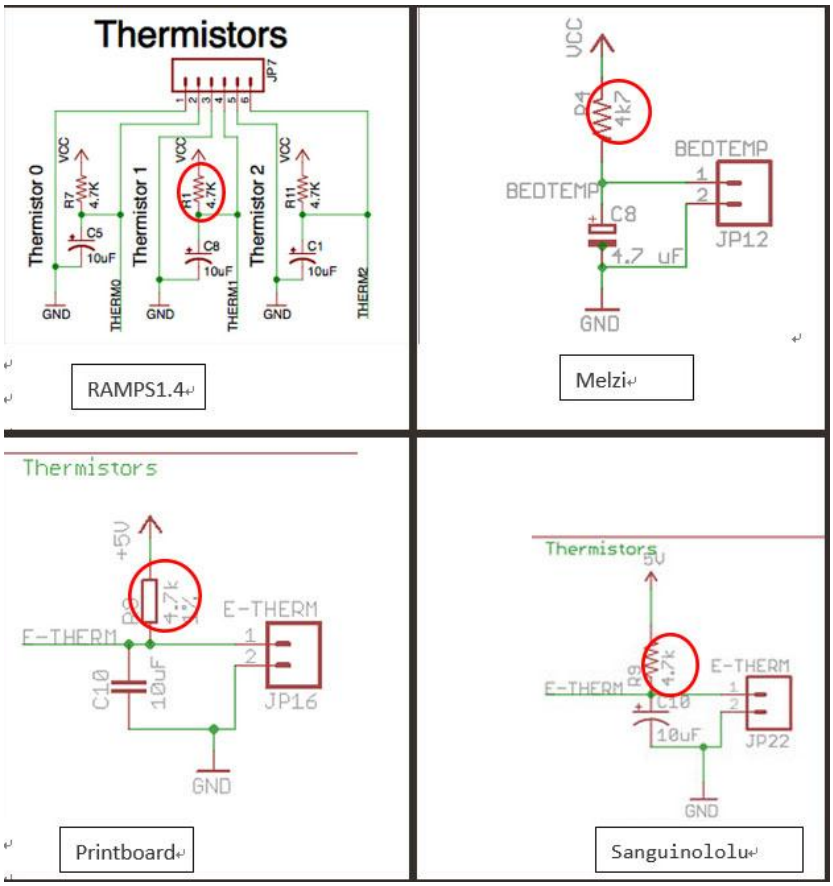

图 1　4.7K 上拉电阻

接下来是温度检测的一些配置，包括双喷头温度差，M109 检测配置，安全温度配置。下面笔者一一解释。

首先下面这一句配置双喷头温差最大值，如果温度超过这个数值，那么打印机会终止工作，因此对于双喷头打印机玩家来说，这个参数需要注意。

```
#define MAX_REDUNDANT_TEMP_SENSOR_DIFF 10
```

下面这一段配置 M109 指令完成的指标，我们知道，M109 指令设定喷头温度并等待，那么等待到什么时候呢？下面这三个参数控制这个时间。第一个参数表示温度“接近”目标温度必须持续 10 秒才算加热完成，第二个参数表示和目标温度相差不超过 3° 为“接近”，第三个参数表示从温度与目标温度相差不超过 1 度开始计时，从此刻开始，温度和目标温度持续接近 10 秒钟，则完成加热。

```
#define TEMP_RESIDENCY_TIME 10
#define TEMP_HYSTERESIS 3
#define TEMP_WINDOW     1
```

下面配置安全温度范围的下限和上限，包括各个喷头和加热床。如果温度超过下限，那么打印机会抛出 MINTEMP 的错误并终止工作，如果超过上限，那么打印机抛出 MAXTEMP 的错误并终止工作。Marlin 用这种方式保护 3D 打印机。下面的配置最小温度都是 5°，喷头的最大温度为 275°，热床的最大温度为 150°。

```
#define HEATER_0_MINTEMP 5
#define HEATER_1_MINTEMP 5
#define HEATER_2_MINTEMP 5
```

```
#define BED_MINTEMP 5
#define HEATER_0_MAXTEMP 275
#define HEATER_1_MAXTEMP 275
#define HEATER_2_MAXTEMP 275
#define BED_MAXTEMP 150
```

如果希望 M105 指令在报告温度的时候，也报告喷头和加热床的功率，则可以将下面两句的前面的"//"去掉。具体的功率数值需要用户自己计算得到。

```
#define EXTRUDER_WATTS (12.0*12.0/6.7)
#define BED_WATTS (12.0*12.0/1.1)
```

接下来配置温度控制方法，Marlin 提供两种温度控制方法，一种是简单的 bang-bang 控制，这种控制方法比较简单，效果较差，另一种是 PID 控制，即比例-积分-微分控制方法，这种控制效果比较好。因此笔者使用 PID 控制。而关于 PID 控制的详细资料，请自行查阅。关于 PID 参数的设置，对普通 3D 打印机玩家来说影响不是很大，一般的参数设置都能满足温度控制的需要，因此使用默认的 Ultimaker PID 参数即可。对于加热床来说，使用默认的控制方法即可。

```
#define PIDTEMP
```

越过了温度控制方法之后，就到了保护挤出机的配置，包括防止冷挤出和过长距离的挤出。防止冷挤出就是在喷头温度低于某个温度的时候是挤出动作无效，而过长距离的挤出是指一次挤出的距离不能大于某个长度。第一句是防止冷挤出，第三句是定义冷挤出的温度，即 170°，玩巧克力或食品打印机的朋友需要注意到这个温度值。第二句是防止冗长挤出，第四局指明了这个距离的数值，为 X 轴长度与 Y 轴长度之和。

```
#define PREVENT_DANGEROUS_EXTRUDE
#define PREVENT_LENGTHY_EXTRUDE
#define EXTRUDE_MINTEMP 170
#define EXTRUDE_MAXLENGTH (X_MAX_LENGTH+Y_MAX_LENGTH)
```

接下来的一大段是为了防止温度失控遭造成着火而设置的，笔者就遇到过这样的情况，当时热敏电阻没有放到喷头上，然后一直在加热，最终将 PEEK 烧爆炸了。这个配置的具体原理是如果测得温度在很长一段时间内和目标温度的差大于某个数值，那么打印机会自动终止，从而起到保护打印机的效果。Marlin 默认将这几句注释掉了，即不做这样的保护，如果用户希望做这样的保护，只需要将注释取消即可。第一句是配置检测时间，第二句是控制温度差距。对于加热床也有类似的配置，需要注意的是，当前越来越大的热床被使用，导致加热速度很慢，要防止被误查。

```
#define THERMAL_RUNAWAY_PROTECTION_PERIOD 40
#define THERMAL_RUNAWAY_PROTECTION_HYSTERESIS 4
#define THERMAL_RUNAWAY_PROTECTION_BED_PERIOD 20
#define THERMAL_RUNAWAY_PROTECTION_BED_HYSTERESIS 2
```

终于来到了机械配置部分，首先需要配置的是限位开关。一般的配置是，对所有的限位开关都使用上拉电阻，而机械式限位开关连接在常闭段，那么限位开关在正常情况下（未触发），信号端（SIGNAL）为低电位，限位开关触发时，开关处于开路状态，信号端为高电位。这和 Marlin 中默认的限位开关逻辑相同。

首先保持使用上拉电阻，

```
#define ENDSTOPPULLUPS
```

接着就是所有的限位开关都使用上拉电阻形式。

```
#ifdef ENDSTOPPULLUPS
  #define ENDSTOPPULLUP_XMAX
  #define ENDSTOPPULLUP_YMAX
  #define ENDSTOPPULLUP_ZMAX
  #define ENDSTOPPULLUP_XMIN
  #define ENDSTOPPULLUP_YMIN
  #define ENDSTOPPULLUP_ZMIN
#endif
```

下面就是限位开关的逻辑配置，如果限位开关连接方式为 GND 端连接限位开关的 COM 端，而 SIGNAL 端连接的是限位开关的常闭（NC）端，那么就把该限位开关对应的逻辑设置为 false，否则设置为 true。如果使用的打印机只有最小值处的限位开关，那么保持默认设置即可。

```
const bool X_MIN_ENDSTOP_INVERTING = false;
const bool Y_MIN_ENDSTOP_INVERTING = false;
const bool Z_MIN_ENDSTOP_INVERTING = false;
const bool X_MAX_ENDSTOP_INVERTING = true;
const bool Y_MAX_ENDSTOP_INVERTING = true;
const bool Z_MAX_ENDSTOP_INVERTING = true;
```

有的打印机并不是使用了 6 个限位开关，大多数情况下，都是使用 3 个最小值处的限位开关，而最大值处的限位开关都没有使用。Marlin 固件允许用户指定所使用的限位开关。下面两行可以选择去告诉打印机没有使用哪些限位开关，笔者打印机只是使用了全部 3 个最小值处的限位开关，因此禁用最大值处的限位开关。即将第一行的注释符“//”去掉。

```
#define DISABLE_MAX_ENDSTOPS
//#define DISABLE_MIN_ENDSTOPS
```

下面配置步进电机的运动方式，主要配置步进电机的正向反向，根据实际情况改变配置即可，如果发现某个步进电机运动方向不对，把对应的配置改为相反的值即可。

```
#define INVERT_X_DIR true
#define INVERT_Y_DIR false
#define INVERT_Z_DIR true
#define INVERT_E0_DIR false
```

紧接着配置回归初始位位置，-1 表示初始位置为坐标最小值处，1 表示初始位在坐标最大值处。笔者的打印机配置如下：

```
#define X_HOME_DIR -1
#define Y_HOME_DIR -1
#define Z_HOME_DIR -1
```

接下来是关于 Marlin 如何确定步进电机已经达到边界的位置，软限位的方式是通过判

断喷头的坐标值是否越过打印机范围，否则根据限位开关的状态判断是否越位。因为笔者的打印机限位开关都在最小值处，为了节省计算资源，只需要对最大值使用软限位方式。配置如下：

```
#define min_software_endstops false
#define max_software_endstops true
```

那么为了正确判断喷头是否越位，需要正确配置打印机的打印范围。MAX 为最大坐标，MIN 为最小坐标。笔者的打印机范围为 200×200×160mm，因此配置如下：

```
#define X_MAX_POS 200
#define X_MIN_POS 0
#define Y_MAX_POS 200
#define Y_MIN_POS 0
#define Z_MAX_POS 160
#define Z_MIN_POS 0
```

接下来的一大段控制自动调平，笔者认为当前 3D 打印机都存在一个平台不平整的问题，因此自动调平没太大作用，因此没有使用自动调平，确保下面一句处于注释状态。

```
//#define ENABLE_AUTO_BED_LEVELING
```

接下来就该配置步进电机的运动选项了。首先是定义轴的数量，对于单喷头机器，应该是 4 轴，分别是 X 轴、Y 轴、Z 轴和 E 轴。然后是回归初始位的速度，注意单位是毫米每分钟。Z 轴回归初始位的速度比较慢，E 轴不存在初始位，因此设置为 0。

```
#define NUM_AXIS 4
#define HOMING_FEEDRATE {50*60, 50*60, 4*60, 0}
```

接下来配置很重要的四个参数，每个轴的运动分辨率，即每个轴方向上发生 1 毫米的运动，对应的步进电机应该转动多少步。一般来说 X 轴和 Y 轴都是步进电机+同步带结构，Z 轴为步进电机+丝杆结构，而挤出机为步进电机+齿轮结构，这四个参数可以通过 Repetier-Host 软件中的计算器（工具菜单中）计算得来。

对于 X 轴和 Y 轴来说，计算原理为步进电机转一周为 360°，与此同时，同步轮发生一周的转动，假如同步轮为 17 齿，那么同步带上一点就运动了 17 个齿距的长度，如果使用同步带齿距为 2 毫米，那么就发生 34 毫米的运动。而假如步进电机步距角为 1.8°，同时驱动器的细分数为 1/16，那么步进电机转一圈就发生 360/1.8*16=3200 步。因此 X 轴和 Y 轴的分辨率就是 3200/34 = 94.12。

对于 Z 轴来说，步进电机转一周，同样转动了 3200 步，而假如使用的丝杆导程为 8 毫米，即丝杆转一圈，丝杆螺母运动 8 毫米，那么 Z 轴的分辨率就是 3200/8=400。

而对于挤出机来说，如果为近端挤出，不需要加减速器，那么步进电机转一周，带动挤出齿轮转一周，那么耗材就被挤出“挤出齿轮的周长”这个距离，假如挤出齿轮直径为 10 毫米，那么 E 轴分辨率就是 3200/(10*3.14) = 101.86。如果挤出机电机带有减速器，这个数值还要除以减速比。

因此笔者配置如下：

```
#define DEFAULT_AXIS_STEPS_PER_UNIT   {94.12,94.12,400,101.86}
```

剩下的就是最大速度及加速度的配置，一般来说，使用默认值即可。如果发现打印机抖动很厉害，可能是因为加速度过大的原因，可以将第二行中的前两个数值改为 3000，把三

行的默认加速度数值改为 1000。

```
#define DEFAULT_MAX_FEEDRATE          {500, 500, 5, 25}
#define DEFAULT_MAX_ACCELERATION      {9000,9000,100,10000}
#define DEFAULT_ACCELERATION          3000
#define DEFAULT_RETRACT_ACCELERATION  3000
#define DEFAULT_XYJERK                20.0
#define DEFAULT_ZJERK                 0.4
#define DEFAULT_EJERK                 5.0
```

笔者使用的主板是 RAMPS1.4，可以选用很多脱机智能控制器，即可以不用联机，使用显示屏里面的菜单就可以控制打印机。笔者使用的是 LCD12864 控制器，因此将下面一句的注释符去掉，告诉固件使用这款控制器。

```
#define REPRAP_DISCOUNT_FULL_GRAPHIC_SMART_CONTROLLER
```

另外一款比较常用的控制器是 LCD2004 控制器，使用这一款就需要把下面一句的注释去掉，并把其他的控制器选项都注释。用户也可以选择其他的控制器，只需做相应的配置即可。

```
#define REPRAP_DISCOUNT_SMART_CONTROLLER
```

智能控制器都有预热菜单，即选择预热 PLA 和预热 ABS，具体的温度可以进行更改。笔者打印 PLA，一般喷头温度设置为 210°，而加热床温度设置为 40°；打印 ABS 的话，喷头温度设置为 230°，加热床温度设置为 60°。因此配置如下：

```
#define PLA_PREHEAT_HOTEND_TEMP 210
#define PLA_PREHEAT_HPB_TEMP 40
#define PLA_PREHEAT_FAN_SPEED 255
#define ABS_PREHEAT_HOTEND_TEMP 230
#define ABS_PREHEAT_HPB_TEMP 60
#define ABS_PREHEAT_FAN_SPEED 255
```

至此，Marlin 基本配置已经完成。可以通过 Arduino IDE 选择相应的端口和板子类型，然后编译上传到主板上，可以通过 Repetier-Host 或者 PrintRun 软件调试打印机，发现问题再调整固件的参数，重新上传，直到打印机正常工作。

后记

Marlin 固件非常强大，很多高级的功能值得 3D 打印机玩家挖掘，笔者也会逐步研究并介绍 Marlin 固件中的基本功能，和大家一起交流。

文档中可能有些错误，欢迎大家指正交流。