

Smoothieware 使用笔记

撰写: DerekYang QQ: 110458952, 1561000324

本人第一次使用这个主控板以及其处理器,有很多不明白的东西。由于也不是专业搞 3D 打印机的只是业余爱好,所以写下来方便查阅。同时也希望给和我一样刚入门的朋友一个参考,也希望高手指点。本文当前只是随手记录,有不正确的地方,请帮助我改进,多多包涵!谢谢!

本文不完善, **请大家不要随意传播**, 免得误导他人, 自己看看给我提意见就好。

MKS SBASE 板子购买于淘宝店 [创客基地](#), 基于开源项目改版而来, 兼容开源固件。

宝贝链接: <https://item.taobao.com/item.htm?id=44718074273>

基本操作和配置, 创客基地自带的文档说明都有, 我就不再重复了, 只将我觉得有必要记录的内容整理一下。

1 开源项目

1.1 简介

Smoothieware 是一个开源项目, 基于 32 位 100M 主频的 LPC17xx 系列处理器。

主要应用于 FDM 3D 打印、激光切割、激光雕刻、CNC 加工等领域。

开源官方网址: <http://smoothieware.org/>

1.2 开源硬件

由于原始图纸较大, 不方便在文本中展示, 请移步到官方查看。(可能需要翻墙)

1.2.1 开源硬件原理图地址 (图片)

<http://smoothieware.org/smoothieboard-schematic>

1.2.2 元件清单

<https://docs.google.com/spreadsheets/d/1vshB97WWt6Lceo59aggbdhRuXxTEhSL9lkotScR0oYE/edit#gid=0>

1.2.3 连接器清单

https://docs.google.com/spreadsheet/ccc?key=0Api7_ZbfikkKdHR5VkdXMFfwcHRF0G5CXzdp0GhqM3c#gid=0

1.2.4 源文件

<https://github.com/Smoothieware/Smoothieboard>

PCB 源文件: smoothieboard-5driver.brd
SCH 源文件: smoothieboard-5driver.sch
设计软件: Cadence

1.3 开源固件

下载地址: <https://github.com/Smoothieware/Smoothieware>

2 固件配置

2.1 固件配置方法

2.1.1 配置文件

固件的配置是动态的,在板载的 TF 卡槽中必须插入 TF 卡。配置参数存储在 TF 卡根目录上的 Config.txt 文件中。根据不同类型的 3D 打印机有默认配置模板供选择和修改,以加快应用。

注意: 开机前几秒,系统初始化啊,并读取 TF 卡中的 Config.txt 文件。所以设计产品时前几秒的 IO 状态需要留意。

2.1.2 M500 保存参数

使用特定的指令可以临时的修改系统参数,如果需要存储改设置,可以使用 M500 指令来将参数存储到 TF 卡上。

问题: M500 指令提示有存盘,但是实际上并没有

问题解决:

该系统不允许 USB 和固件同时访问 TF 卡,需要在计算机上弹出大容量存储设备,之后使用 M500 指令,即可存盘。

关于存盘: 不会修改 config.txt 文件,会生成一个 config-override 文件,文件中以 M 指令的方式覆盖 config.txt 中的设置。也就是系统先加载 config.txt 文件的配置,然后检查是否存在 config-override 文件,如果存在,再执行该文件中的 M 指令再次定义参数。

2.1.3 config-get/set 指令

系统提供在线修改 config.txt 中的参数的指令。(控制台指令需要在以“@”开头)

注意,操作 TF 卡同样需要弹出大容量存储设备!

例如: @config-get sd acceleration 获得加速度参数

```
>>>@config-get sd acceleration
SENDING:config-get sd acceleration
sd: acceleration is set to 3200
```

或 @config-set sd acceleration 3000 来设置加速度

```
>>>@config-set sd acceleration 3000
SENDING:config-set sd acceleration 3000
sd: acceleration has been set to 3000
```

2.2 固件引脚配置

示例: `alpha_max_endstop 1.25^`

引脚编号后面的修饰符定义如下:

- ! 逻辑非
- 0 引脚开漏
- ^ 引脚上拉
- v 引脚下拉
- 无上拉或下拉
- @ 中继模式

2.3 固件配置选项

参数众多, 请查阅官方: <http://smoothieware.org/configuration-options>

2.4 命令控制台

本系统支持命令控制台, 详细信息请参阅: <http://smoothieware.org/console-commands>

SimpleShell 是一个小型 Unix 控制台模块, 使您可以浏览文件系统 (SD 卡或其他) 并显示这些文件的内容。

常用: `version help ls cd cat play progress abort mem break net rm pwd`

2.5 常用命令

指令	说明	示例
M301	设置温控 PID 参数	M301 S0 P30 I10 D10
M305	设置 NTC 参数	M305 S0 B3950 R100000 T25
M500	保存临时设置的参数	
M501	加载 config-override	M501 test1 说明: 加载 config-override.test1 文件
M502	删除配置覆盖文件	
M503	回显 config-override 文件内容	
M504	保存覆盖文件选择不同的扩展名	M504 blue-pla 说明: 保存当前内存中的参数到 config-override. blue-pla

3 温度及控制

温度控制的官方参考页面: <http://smoothieware.org/temperaturecontrol>

3.1 配置 NTC 温度传感器

3.1.1 使用 3 点设置法

在配置文件相应位置添加项目 (例如加热头)

```
temperature_control.hotend.rt_curve 25,100000,150,1665,240,266.2
```

NTC 100K B3950 的 25, 150, 240 度时的电阻值

3.1.2 参数设置法

用 M 代码设置

```
M305 S0 B3950 R100000 T25
```

3.1.3 Steinhart Hart 参数法

官方原档认为这种方法是最准确的。

每个 NTC 需要 3 个参数: I Steinhart, J Steinhart, K Steinhart。可由固件自动计算。

自动计算示例如下:

```
@calc_thermistor 25,100000,150,1665,240,266.2
```

从返回值中可以看到计算得到的 3 个值:

```
Steinhart Hart coefficients: I0.000744991004467010 J0.000212162165553309 K0.000000109054624886
```

Paste the above in the M305 S0 command, then save with M500

或 写入配置文件:

```
temperature_control.hotend.coefficients 0.000744991004467010,0.000212162165553309,0.000000109054624886
```

或 加 “-s0” 参数, 表示为 s0 计算这个值, 注意小写 s

```
@calc_thermistor -s0 25,100000,150,1665,240,266.2
```

```
Steinhart Hart coefficients: I0.000744991004467010 J0.000212162165553309 K0.000000109054624886
```

Setting Thermistor 0 to those settings, save with M500

3.2 配置热电偶温度传感器

热电偶的设置暂时没有测试

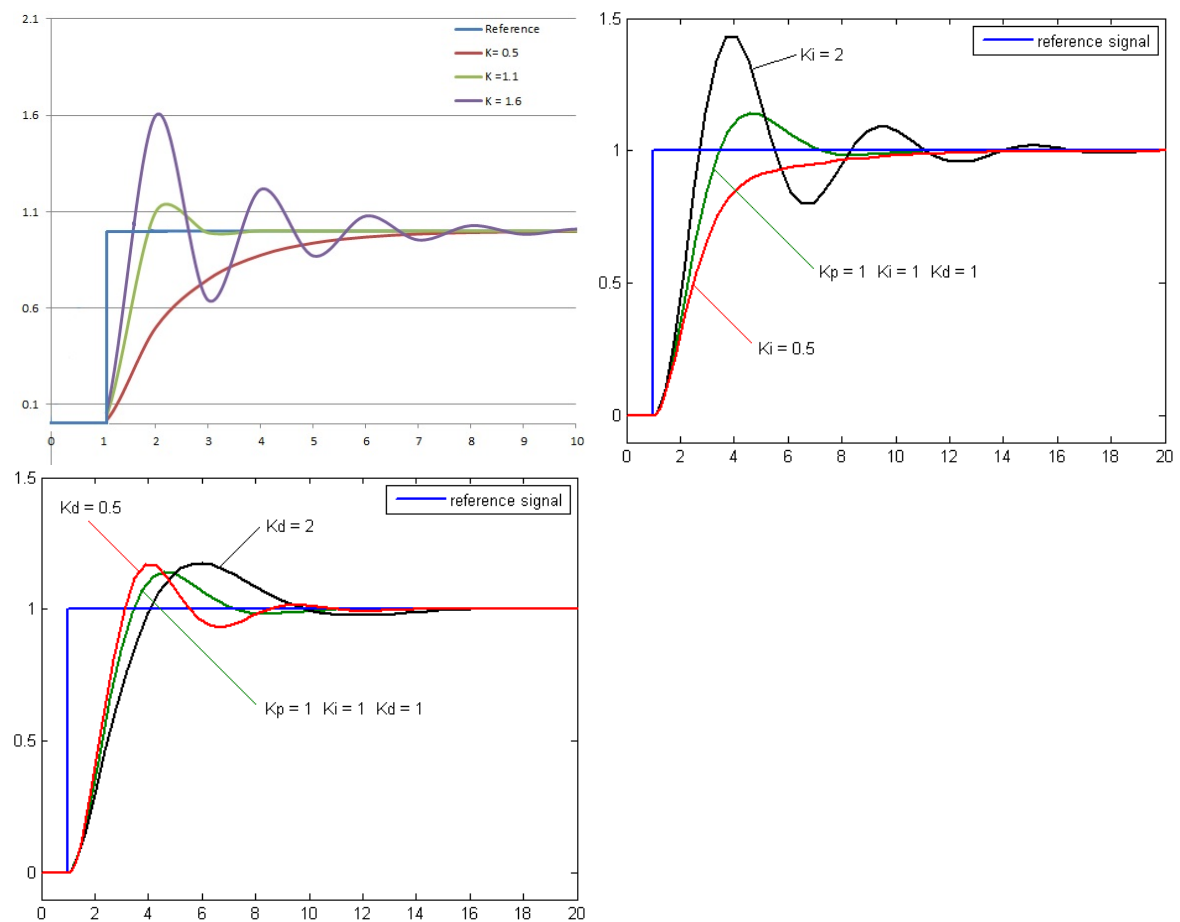
3.3 PID 温度控制方法

3.3.1 PID 温控介绍

维基百科对 PID 控制算法有详细介绍

网址: https://en.wikipedia.org/wiki/PID_controller

另 PID 动画: https://en.wikipedia.org/wiki/File:PID_Compensation_Animated.gif



3.3.2 参数设置

设定 PID 参数

```
M301 S0 P30 I10 D10
```

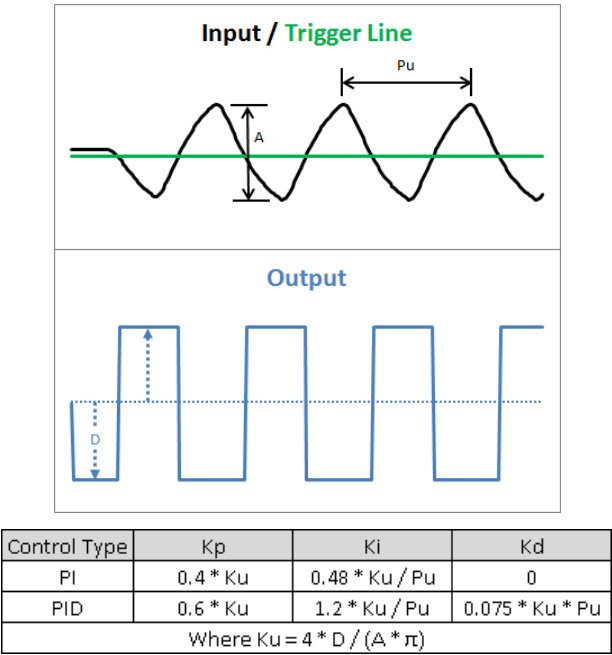
这个设定只在内存中生效，并没写到配置文件中

用 **M500** 指令写入到配置文件中

3.3.3 参数自整定

PID 自整定算法参考：（这里有详细的测量和计算方法）

<http://brettbeauregard.com/blog/2012/01/arduino-pid-autotune-library/>



对挤出头温控进行自整定

指令：

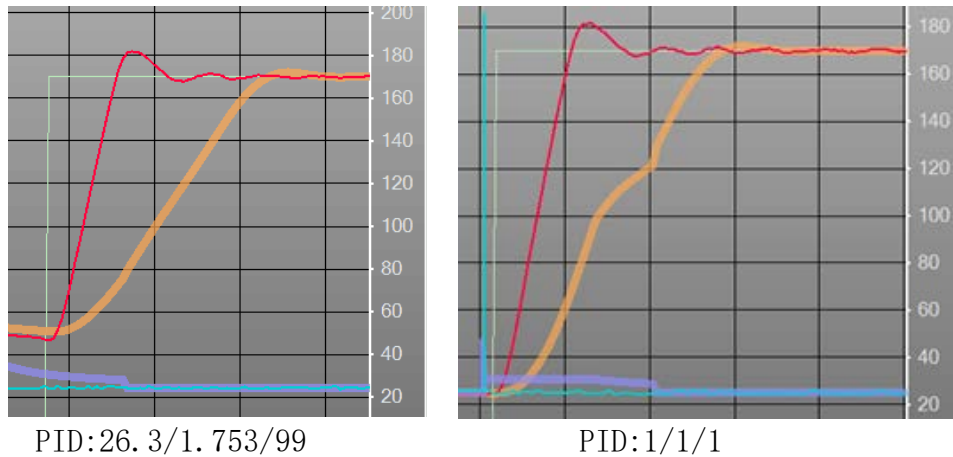
```
M303 E0 S190
```

解释：对 E0 按 190 度进行自整定

自整定过程会将温度升至 190 度，并在 190 度上下震荡几个周期，计算得到如下应答：

```
00:18:17.232 : Cycle 4: max: 202.865, min: 188.072, avg separation: 0.362854
00:18:17.233 : Ku: 43.8957, Pu: 30.05
00:18:17.233 : Trying:
00:18:17.233 : Kp: 26.3
00:18:17.233 : Ki: 1.753
00:18:17.233 : Kd: 99
00:18:17.234 : PID Autotune Complete! The settings above have been loaded
into memory, but not written to your config file.
```

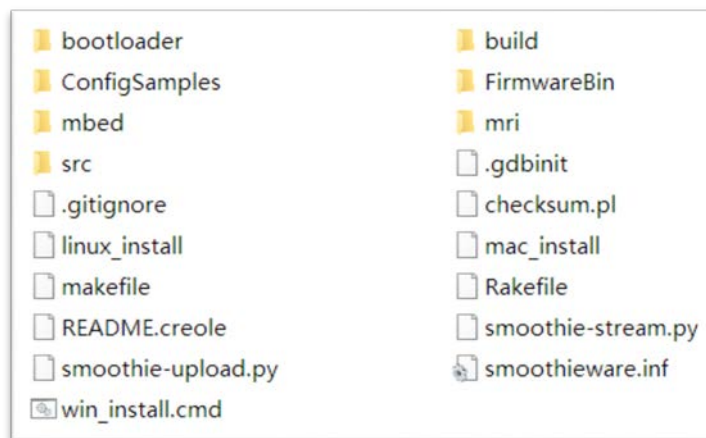
以我的 Kossel 机器为例，整定后设定温度 170 摄氏度测试的控制结果如下：



4 固件

4.1 固件源码

固件压缩包 **Smoothieware-edge.zip**，压缩包内容如下图：



4.1.1 编译环境及安装

编译环境：GCC ARM Embedded

版本：V4.8

官方网址：<https://launchpad.net/gcc-arm-embedded/>

安装

运行 win_install.cmd 安装 Windows 编译器运行环境，根据提示进行安装。

安装主要是下载 **gcc-arm-none-eabi-4_8-2014q1-20140314-win32.zip** 文件，然后解压，然后配置一些东西（我还没有弄明白，也不需要弄明白）。

注：这里没有用最新版本，因为我也不清楚是不是能直接用最新版。由于安装时下载这个文件的服务

器在国外，最好使用代理，否则安装程序下载到一半的时候就失败了。我就是试了好几次。

4.1.2 编译固件源码

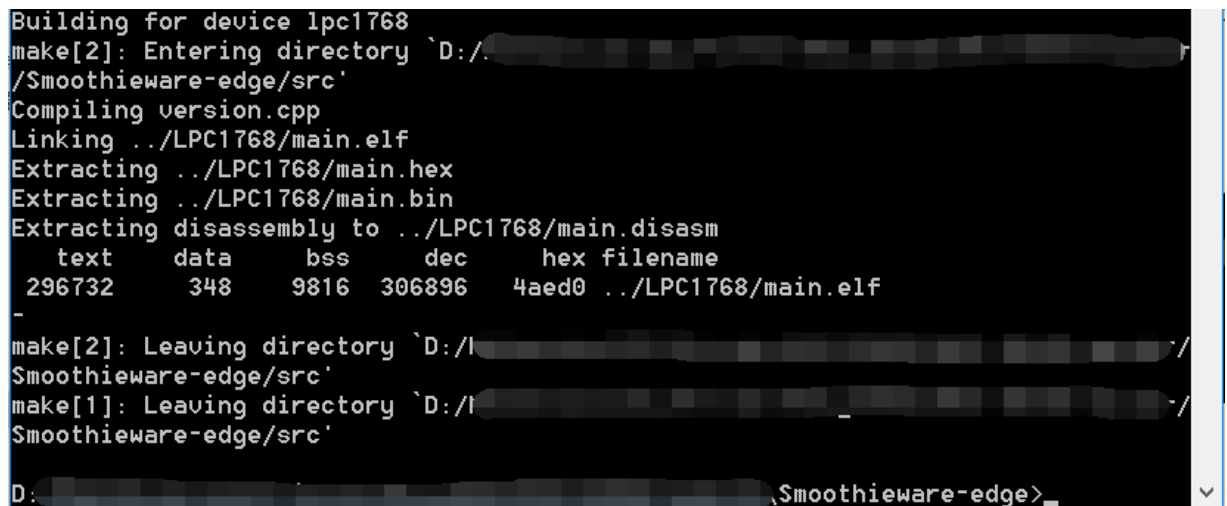
安装好编译环境后，可以直接运行一下，看看是否能正确的编译。

由于下载的固件包已经都配置好了编译选项，我暂时不去管编译选项的内容，直接可以编译。

打开编译环境：运行 BuildShell.cmd



直接输入 make 回车就开始编译了。



```
Building for device lpc1768
make[2]: Entering directory `D:/Smoothieware-edge/src'
Compiling version.cpp
Linking ../LPC1768/main.elf
Extracting ../LPC1768/main.hex
Extracting ../LPC1768/main.bin
Extracting disassembly to ../LPC1768/main.disasm
  text    data    bss    dec    hex filename
 296732   348   9816 306896 4aed0 ../LPC1768/main.elf
-
make[2]: Leaving directory `D:/Smoothieware-edge/src'
make[1]: Leaving directory `D:/Smoothieware-edge/src'
D:\Smoothieware-edge>
```

(编译完成)

编译结束，编译的结果，对于我们最重要的就是生成了一个 bin 文件：/LPC1768/main.bin

4.1.3 下载固件

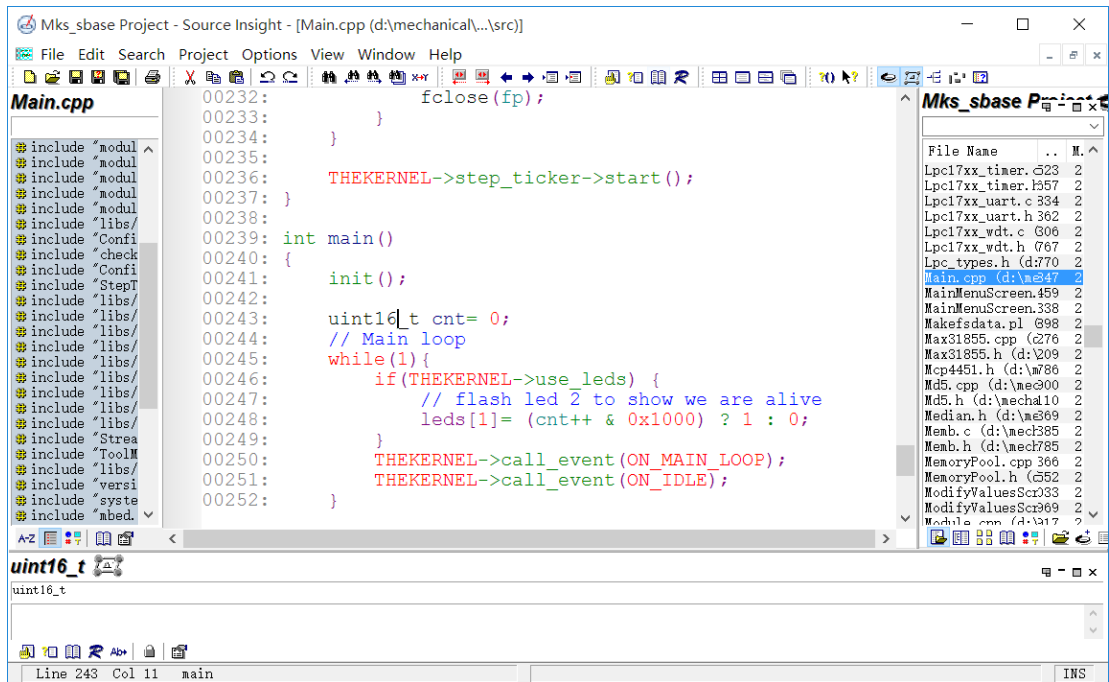
将上节生成的文件改名为：firmware.bin，然后拷贝到板子上的 TF 卡内。重新开机，等待几秒。程序重新运行起来表示挂件更新成功。

4.2 修改固件

4.2.1 准备

我还不知道有没有 IDE 环境来专门做这个的编译调试，所以我直接用 Source Insight 建立一个工程，并加入所有源文件。

源文件为 src 目录及子目录下的所有文件。



4.2.2 修改 LCM12864 屏旋钮动作

因为原版固件处理这个旋钮不好操作。表现为：旋转一格动作不止一格，菜单滚动 2 个项目，数值加减也不止一个数，需要旋转半格才能实现一个动作。操作很不方便。所以我首先修改这个。

因为旋钮的触发信号大致是转一格，触发了几次，所以要减少触发量。

打开文件：src\modules\utils\panel\panels\ReprapDiscountGLCD.cpp，找到下面的函数：

```

int ReprapDiscountGLCD::readEncoderDelta() {
    static const int8_t enc_states[] = {0,-1,1,0,1,0,0,-1,-1,0,0,1,0,1,-1,0};
    static uint8_t old_AB = 0;

    old_AB <= 2; //remember previous state
    old_AB |= ( this->encoder_a_pin.get() + ( this->encoder_b_pin.get() * 2 ) ); //add current state
    return enc_states[(old_AB&0x0f)];
}

```

修改为：

```

int ReprapDiscountGLCD::readEncoderDelta() {
    static const int8_t enc_states[] = {0,-1,1,0,1,0,0,-1,-1,0,0,1,0,1,-1,0};
    static int8_t encoder_fp = 0;
    static uint8_t old_AB = 0;
    old_AB <= 2; //remember previous state
    old_AB |= ( this->encoder_a_pin.get() + ( this->encoder_b_pin.get() * 2 ) ); //add current state
    encoder_fp += enc_states[(old_AB&0x0f)];
    if(encoder_fp >= 2)
    {
        encoder_fp = 0;
        return 1;
    }
    else if(encoder_fp <= -2)
    {
        encoder_fp = 0;
        return -1;
    }
    return 0;
}

```

编译固件->下载固件->重新开机->测试 OK

5 其他问题

5.1 加热棒兼容性

如果 24V 系统使用 12V 的加热棒，需要配置下面选项，来限制最大供电百分比。（默认 255）
 temperature_control.hotend.max_pwm 64 #限制最大加热比例，最大 256