电机程序使用文档

# 测试用BIN文件使用说明

如果只使用VEMotorTest.bin文件，你可以测试电机的正转反转和LED的灯光色彩。

向芯片内烧入VEMotorTest.bin文件。启动后按SW2时电机顺时针旋转，按SW3时电机逆时针旋转，同时按下时LED会有七彩灯光变化。

# 成品程序使用说明

程序从UART、CAN、WIFI三种接口接受数据。UART是电路板最左边的一列插口，从上到下依次是 GND Rx Tx VCC。UART接口不仅接受用户命令，而且输出日志信息。

如果你想要调整程序的设置，方式有从源码设置、使用设置命令、或者在Flash的对应地址写入设置内容的方法。源码设置需要在IDE里面导入项目更改代码之后重新编译。源码设置的内容表示当尝试读取Flash设置储存区的时候，如果对应地址没有数据，那么会使用的默认值。当Flash里面有设置的时候使用Flash里面读到的数据。当你使用设置命令的时候，设置数据会保存在Flash里面，所以机制是先读取Flash，如果里面有数据那么不会使用源码里写的默认值。

Flash里储存的设置，在使用IDE重新烧入芯片程序的时候，有可能被清空，因为IDE在烧写程序的时候擦除的范围不是不变的。如果你使用烧写器烧入程序，不要覆盖Flash地址256KB之后的内容，设置就不会被清空。

如果你不更改C语言代码里的设置默认值，而是选择使用设置命令写入设置，或者在烧入的时候通过烧录器在对应地址写入数据的话，那么你可以直接使用 VEMotor.bin 文件，这个文件从0x0800 0000也就是Flash的起始地址烧入，设置数据的烧入位置之后会有介绍。

# 启动过程

在系统启动时候首先报告它从Flash里面读取到的数据，或者读取不到数据而使用的默认值。之后根据config.h里面的设置，如果启用电机归位初始化，那么会把电机回卷收入盒内，如果十秒之后仍然得不到限位开关的反馈，那么会判定超时，停止电机移动，如果得到了限位开关反馈，那么会把这个位置定为位置零点；如果在代码里面启用了WIFI功能模块，那么在启动的时候首先会等两秒钟让ESP-01S模块先完成启动，然后如果代码里面设置在初始化时连接TCP Server那么会尝试连接。整个过程在UART上有日志记录。

# 日志信息的格式

[Log-Level][Time] Message

日志的行尾是CRLF。

Log-Level有 Trace, Debug, Info, Warn, Error 五种。

Time 是开启启动之后经过的毫秒数。

Message 是输出的日志信息。

# UART/WiFi TCP使用的命令格式

[#<receiver> ]<command>[ <args…>]

方括号的内容表示可选项。

命令的输入行尾可以使用CR或者LF，但是不能使用CRLF，否则会当作两条命令并且后一条命令为空，后一条命令会给出Error的反馈。

Receiver是一个从1-65535的数字，表示这个命令的接受者编号。如果给出Receiver，并且编号不是自己的Machine-id的时候，会把这条命令向CAN总线发送，如果是自己的Machine-id，那么当作没有给出Receiver的命令一样处理。

WiFi相关的命令不能通过CAN发送，传输过去也没有效果。因为CAN只有8个字节的报文，如果要设置SSID请逐个使用UART连接或者在源码或者Flash里面设置好默认值。

每条命令的处理结果都会显示一行反馈，内容是<OK>或者<FAIL>，反馈的行尾是CRLF。在WiFi上面的反馈每个都会装在不同的TCP包里。

如果是通过CAN发送出去的命令，接受方会把处理结果通过CAN总线反馈给发送方，发送方收到反馈结果之后会在UART以及WiFi上面输出一行<#receiver OK>或者<#receiver FAIL>，<receiver>是原先命令里的Receiver数字，行尾是CRLF。所以一条“#124 ms”会出现“<OK>”和“<#124 OK>”这样两句反馈。

如果是和WiFi模块通讯的指令，也就是表格里面以字母w开头的指令（不包括wsetap / wsettcp这两条设置指令），这些需要通讯的指令因为都需要时间，所以反馈的<OK>或者<FAIL>只表示指令是否成功读取，但是不表示是否在WiFi模块上执行成功，是否执行成功要根据日志内容来判断。

在发送多条指令的时候，可以把多条指令写在不同行内一次性发送。UART连接可以连续发送指令，有1KB的缓存区，使用者只要预估一条指令使用10ms不要太频繁即可；WiFi连接的时候，因为无线通讯机制等等问题，使用者应该预估一条指令要用400ms，也就是说如果你一次性在TCP通讯上对一台机器发送10条指令，接下来的4s里面应该保持空闲，等待机器反馈的<OK>/<FAIL>。

拥堵可能在CAN总线，UART传输速度，Wi-Fi拥挤三个层面上发生。

可用的命令列表：

|  |  |  |
| --- | --- | --- |
| 命令名称 | 参数 | 说明 |
| mr | <dir> <speed> | Motor Run电机匀速运动，dir=0时向外，dir=1时向内，speed范围从1到30。匀速运动的时候如果按下限位开关那个位置会被校准成步数0所在的位置。 |
| ms | 无 | Motor Stop电机停止。 |
| mp | <step> | 电机移动到步数，初始化的时候把步数0校准到了最内侧，所以mp 0会移动到最内侧，mp -110000 会移动到最外侧，为了防止撞击到边界，实际移动的时候数字范围在 -5000 ~ -100000 之间。 |
| mpp | <percent> | 电机移动到百分比，percent=0的时候最内侧，percent=100的时候移动到最外侧。实际对应的步数在 config.c 代码的设置部分或者Flash里。 |
| mpr | <step> | 电机移动步数，正数的时候向内，负数的时候向外。例如 mpr -10000 向外移动5cm左右。 |
| mwhere | 无 | 通过UART日志报告现在的电机位置。 |
| mhome | 无 | 把当前位置设置成步数0所在的位置。 |
| mcycle | 无 | 电机循环测试命令。测试的内容在之后的设置部分里说明，进入测试状态之后输入任何一句mcycle之外的命令都会退出测试。 |
| mreset | 无 | 重置L6470步进电机驱动芯片。这在L6470过热或者因大电流进入保护状态时可以使用。这个命令会把目前位置视作步数0所在的位置，之后要使用mr 1 10校准。 |
| msetin | 无 | 把当前步数设置成mpp 0移动到的步数，设置内容会写入Flash保存。 |
| msetout | 无 | 把当前步数设置成mpp 100移动到的步数，设置内容会写入Flash保存。 |
| mid | <machine-id> | 更改Machine-id机器编号，设置会写入Flash保存。 |
| mload | 无 | 重新载入Flash里面保存的所有设置，在UART上会报告读取到的内容，和系统启动时做的一样。 |
| wat | 无 | 向ESP-01S模块发送AT指令，用来检测ESP-01S有没有正常工作，如果看到日志里有WiFi returns OK表示正常。 |
| wauto | 无 | 执行一系列配置操作连接热点连接TCP服务器，如果config.h里面开启了初始化时连接TCP服务器，那么和开机时做的一样。如果遇到问题参考测试章节里面的说明。 |
| wjoin | 无 | 根据设置里的SSID和密码连接WiFi热点，建议直接使用wauto。 |
| wleave | 无 | 断开WiFi连接。 |
| wtcp | 无 | 根据设置里的IP和Port连接TCP服务器，建议直接使用wauto。 |
| wdrop | 无 | 断开TCP连接。 |
| wsetap | <ssid> <pwd> | 设置WiFi热点的名称和密码，SSID不能包含空格，参数不能使用引号包裹。设置内容会写入Flash保存。 |
| wsettcp | <ip> <port> | 设置TCP服务器的IP和端口，参数不能用引号包裹。设置内容会写入Flash保存。 |
| led | <r> <g> <b> | 设置LED颜色，三个参数都是0~255的数字。如果在渐变模式里，这条命令会退出渐变模式。 |
| ledgrad | <r> <g> <b>  <ms> | LED从目前的颜色要渐变到的颜色，前三个参数是0~255的数字，后面一个参数是0~65535的毫秒数。 |
| sid | 无 | 报告自己的编号，会在WiFi或者UART上面输出一行反馈，<machine\_id>，例如“<128>”。在WiFi上它和<OK>会分开在两个TCP包里。 |
| chipreset | 无 | 在1秒后重启芯片。1秒钟是为了回复消息。 |

在Unity的示例程序里面使用的是这样几个命令，可以在 C# 代码里面找到：

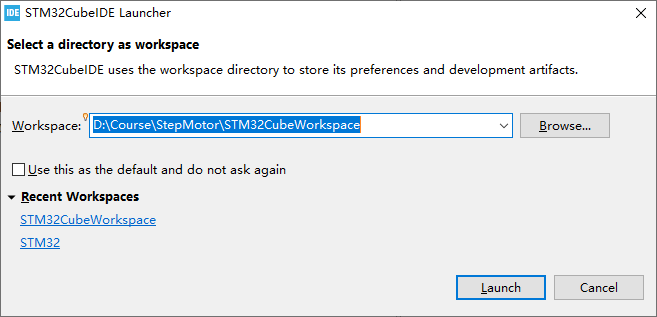
1. mr 1 10 向内以10的速度移动，这个命令用来收回电机。
2. ms 停止电机。
3. mpp <0~100> 对应示例程序里的滑动条，可以鼠标拖动把电机移动到对应位置，注意处理这样的连续输入的时候要滤波并减少命令频率，否则电机运行会非常卡顿。

# 项目设置

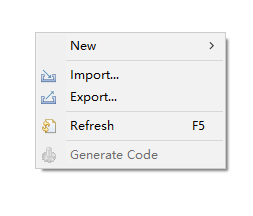
首先介绍从项目代码修改设置的方法，之后会介绍更改Flash数据写入设置的方法。

## 项目源码导入方式

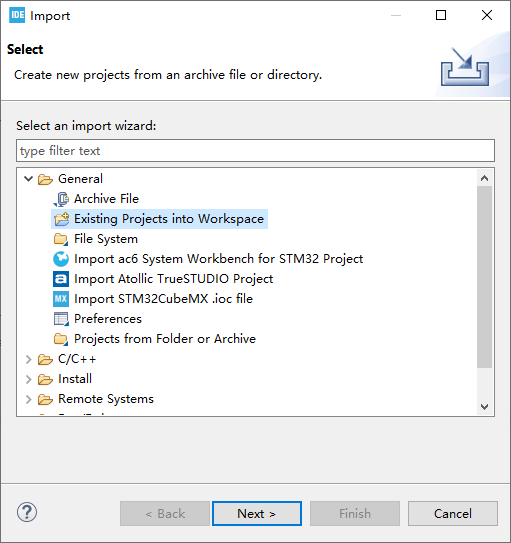
1. 解压之后得到源代码VEMotor，把它放在STM32CubeIDE的Workspace文件夹里面，如果有旧版本的代码那么移除旧版本的代码。Workspace文件夹是指在启动STM32CubeIDE的时候如图让你选择的文件夹。



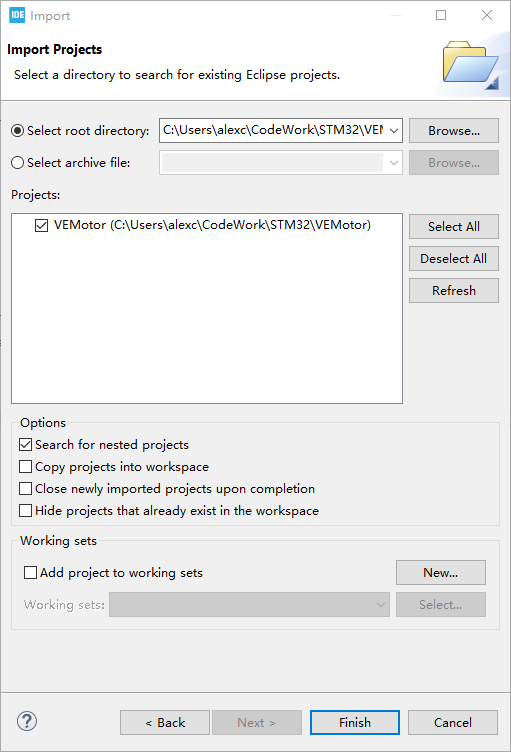
1. 之后启动STM32CubeIDE。
2. 查看界面左侧的项目列表，如果列表里有名叫VEMotor的项目，尝试单击选中之后点击上面的锤子按钮可以编译，那么不需要手动导入。如果没有这样的项目，那么需要手动导入这个项目，手动导入的方法如下：
3. 在项目列表的空白处右键，选择Import…



1. 在导入类型里面选择Existing Projects into Workspace…



1. 点击Next，在下一个窗口里面选择刚刚解压出来的文件夹，名叫VEMotor。



1. 然后点击Finish，在项目列表里面选择新导入的项目，点击菜单栏的锤子图标尝试编译，可以编译就表示导入成功。

### 可能出现的问题：

可能会出现项目使用的固件版本和已经安装的固件版本不一致的提示，这时候需要升级固件版本，应该会有对话框提示这个操作，固件的下载速度翻墙可能更快。

## 源码配置方法

源码配置表示在Flash里面没有设置内容的时候使用的设置默认值。配置项目主要在Src/config.c（下面简称config.c）和Inc/config.h（下面简称config.h）里。配置包括是否启用WiFi，需要的WiFi用户名密码，连接的TCP服务器IP和端口等等。

1. 连接在一个CAN总线上的不同机器需要有不同的ID才能正常通信，ID的设置位置在config.c里面：

// 这个是机器编号，范围是1~65535。

// 不能使用 0 作为 id，那是一个广播编号。

uint8\_t machine\_id = 128;

1. 对于没有安装WiFi模块的板子禁用WiFi可以减少报错，在config.h里面修改这一行：

#define WIFI\_ENABLE 1

1. 如果要在在初始化的时候就尝试连接TCP服务器，在config.h里面对应这一行：

#define INIT\_WIFI\_CONNECT 1

1. 为了正确定位，需要在初始化的时候让电机反转收回最内并校准步数，在 config.h 里面对应这一行：

#define INIT\_MOTOR\_MOVE 1

1. WiFi可以开启每十秒发送一次数据以确认连接情况，这个在config.h里面对应这一行：

#define WIFI\_GREET 1

1. WiFi的用户名密码，以及TCP服务器的地址和端口对应config.c里面的这里：

char wifi\_conf\_ssid[WIFI\_STRSIZE] = "ssid";

char wifi\_conf\_pwd[WIFI\_STRSIZE] = "pwd";

char wifi\_conf\_tcpip[WIFI\_STRSIZE] = "192.168.0.103";

uint16\_t wifi\_conf\_tcpport = 5577;

1. 电机按比例移动的命令 mpp [0~100] 使用的步数范围：

int32\_t lm\_conf\_limit\_out = -110000;

int32\_t lm\_conf\_limit\_in = -5000;

1. 电机循环测试使用的设置，在config.c里面：

// 电机循环测试

int lm\_cycle\_out = -100000;

int lm\_cycle\_in = -5000;

// 1的时候只有跑完一整个测试循环，电机在最里面的位置才会暂停一会儿，

// 0的时候在测试里的每一步都会暂停一会儿。

int lm\_cycle\_pause\_at\_full\_cycle = 1;

int lm\_cycle\_step\_pause = 500;

1. 对于不同型号的电机，它们的需要的出力占空比不一样。如果出力占空比比电机需要的数字更大，这个时候电流很大，容易引发芯片过热停转，如果数值太小电机出力很少，容易被外界力量影响导致电机计步的位置错误。调整出力占空比在文件 Lib/DSpin/dspin.c的函数L6470\_Configuration1(void)里面：

/\* 这里要注意不同型号的电机需要的占空比很不一样，需要调整下面的四个数字 \*/

/\*保持占空比（转矩）设定为10％，范围在0到99.6％\*/

dSPIN\_RegsStruct.KVAL\_HOLD = Kval\_Perc\_to\_Par(10);

/\* 运行占空比（转矩）设定为10％，范围在0到99.6％\*/

dSPIN\_RegsStruct.KVAL\_RUN = Kval\_Perc\_to\_Par(10);

/\* 加速的占空比（转矩）设定为15％，范围在0到99.6％\*/

dSPIN\_RegsStruct.KVAL\_ACC = Kval\_Perc\_to\_Par(15);

/\* 减速的占空比（转矩）设定为15％，范围在0到99.6％ \*/

dSPIN\_RegsStruct.KVAL\_DEC = Kval\_Perc\_to\_Par(15);

1. Mp/mpp/mpr命令的速度相关设置在dspin.c里面，最大速度和噪音高度相关：

/\* 加速率的设置为200 steps/s2，范围14.55至59590 steps/s2\*/

dSPIN\_RegsStruct.ACC = AccDec\_Steps\_to\_Par(200);

/\* 减速率的设置为200 steps/s2，范围14.55至59590 steps/s2 \*/

dSPIN\_RegsStruct.DEC = AccDec\_Steps\_to\_Par(200);

/\* 最大速度设置为400步/秒，最大速度设置范围为15.25至15610步/秒\*/

dSPIN\_RegsStruct.MAX\_SPEED = MaxSpd\_Steps\_to\_Par(400);

/\* 最小速度设置为0步/秒，取值范围为0至976.3，步骤/秒\*/

dSPIN\_RegsStruct.MIN\_SPEED = MinSpd\_Steps\_to\_Par(0);

/\* 全步进速度设置400步/秒，范围为7.63到15625步/秒\*/

dSPIN\_RegsStruct.FS\_SPD = FSSpd\_Steps\_to\_Par(400);

## 命令设置方法

设置命令在之前的命令表格里面都有出现，一共有这样几个：mid, msetin, msetout, wsetap, wsettcp, mload。

## 设置项目对应的Flash位置

设置项目里的数字内容使用小端序储存，字符串内容最后需要一个NULL字符结尾。每个设置项之间都间隔1KB，这样可以单独擦除Flash page不影响到别的储存项目。

Flash里储存的设置，在使用IDE重新烧入芯片程序的时候，有可能被清空，因为IDE在烧写程序的时候擦除的范围不是不变的。如果你使用烧写器烧入程序，不要覆盖Flash地址256KB之后的内容，设置就不会被清空。

表格里的位置说的是相对于Flash起始地址0x0800 0000的偏移量。

|  |  |  |  |
| --- | --- | --- | --- |
| 项目名称 | 类型 | Flash位置偏移 | 说明 |
| Machine id | Uint16\_t | 0x40000 (256KB) | 机器编号，批量生产时可以使用滚码方式烧入，范围是1-65535。 |
| WiFi SSID | string | 0x40400 | WiFi热点名称 |
| WiFi Pwd | string | 0x40800 | WiFi热点对应的密码 |
| TCP Server IP | string | 0x40c00 | TCP服务器的IP地址 |
| TCP Server Port | uint16\_t | 0x41000 | TCP服务器的端口 |
| Motor limit out | int32\_t | 0x41400 | msetout 命令修改的对象，mpp 100的步数位置。 |
| Motor limit in | int32\_t | 0x41800 | Msetin 命令修改的对象，mpp 0的步数位置。 |

# 硬件测试方法

1. 如果需要测试WiFi功能，首先在电脑上运行NetAssist.exe，出现网络调试助手，在左上角，选择协议类型 TCP Server，本地主机地址选择的IP和端口数字要和设置内容一样，然后点击“打开”。



1. 如果需要测试LED灯带，四个引脚的定义从左到右分别是：

Blue，Red，Green，+24V

这个是根据我手上的一个24V灯带设计的。如果需要更改引脚顺序的话，在main.c的main() 函数里面有这样一句：

led\_init(&htim1, TIM\_CHANNEL\_2, TIM\_CHANNEL\_3, TIM\_CHANNEL\_1);

这里三个参数分别是RGB三个颜色使用的定时器输出通道，通道1是最左面的针脚，现在TIM\_CHANNEL\_1是第三个参数，表示通道1也就是左面第一个针脚对应Blue，TIM\_CHANNEL\_2是第一个参数表示通道2也就是左面第二个针脚对应Red。只要颠倒这几个参数的顺序就可以更改针脚的颜色定义。

1. 接上串口和ST-Link烧入程序。
2. 按SW2时电机顺时针旋转，按SW3时电机逆时针旋转，同时按下时LED会有七彩灯光变化。
3. 如果WiFi连接没有问题的话，那么在网络调试助手里面可以看到问好，还有每十秒钟一次的心跳。在WiFi下面的发送框里面发送mcycle指令，可以测试电机循环运转。在测试状态里面发送任何指令都会停止测试。
4. WiFi下面发送任何指令会收到<OK>或者<FAIL>的回复，收不到表示没连上。
5. 如果mcycle循环运转里面有不正常的卡顿或者失速，很有可能是芯片过热或者占空比配置不正确导致瞬间电流过大引起驱动芯片的自我保护。代码里面默认的休息时间很短，很容易引发过热保护停两秒钟。
6. 测试LED需要这个指令，led <r> <g> <b> 三个都是0-255之间的数字。可以通过串口或者WiFi输入。

## 可能出现的问题：

如果WiFi有问题，看串口里面的Error输出。如果它说WiFi returns FAILED 或者 WiFi returns TIMEOUT的话，检查config.c里面的WiFi配置是否正确。如果还是不行，检查启动的时候自动连接WiFi的部分（这个需要前面设置章节里的这一句 #define INIT\_WIFI\_CONNECT 1），到底错在哪里了。可能会有这样的内容：

[Error][18240] WiFi Auto Setup task list failed

[Error][18240] Failed on WiFi connect AP, check your router and WiFi SSID/PWD settings

[Debug][18240] WiFi returns Timeout on index 6 of Auto Setup.

表示它连不上热点。

或者：

[Error][5280] WiFi Auto Setup task list failed

[Error][5280] Failed on TCP connect, check your TCP server and IP/Port settings

[Debug][5280] WiFi returns Error on index 7 of Auto Setup.

表示它连不上TCP服务器。

如果输入命令mcycle出现电机在轨道上原地抖动，这个表示电机连接异常，可能有一根线接触不好。