Motor Control Shield 用户手册

产品概述

本产品是基于 Arduino/ NUCLEO 的点击驱动扩展板，板载双桥驱动芯片 L293D，能够同时驱动四路直流电机或两路步进电机。

提供 Arduino 和 STM32 控制例程

产品特性

- 可同时驱动四路直流电机或两路步进电机
- 电机驱动电源可通过调休配置为 5V 或可调电源
  - 使用 5V 电源驱动电机时，可同时驱动 4 路 5V 直流电机
  - 使用可调电源驱动电机时，可同时驱动两路 1.25V-6.45V 直流电机
- 采用双桥驱动芯片 L293D
  - 该芯片可同时驱动两路直流电机或一路步进电机
  - 4 路 H 桥，每路 H 桥输出电流为 600mA，峰值电流可达 1.2A
  - 自带 ESD 保护

管脚配置

Motor Control Shield	Arduino	NUCLEO	直流电机	步进电机
3V	3V	3V	无	公共端
M1A	D2	PA10	直流电机 1	步进电机 1
M1B	D3	PB3		
M2A	D4	PB5	直流电机 2	
M2B	D5	PB4		
M3A	D7	PA8	直流电机 3	步进电机 2
M3B	D8	PA9		
M4A	D12	PA6	直流电机 4	
M4B	D13	PA5		

M1EN	D6	PB10	M1A, M1B 使能控制
M2EN	D9	PC7	M2A, M2B 使能控制
M3EN	D10	PB6	M3A, M3B 使能控制
M4EN	D11	PA7	M4A, M4B 使能控制

注：M1EN、M2EN、M3EN、M4EN 在模块中未印丝印，它们分别是两个 L293 驱动芯片的硬件，将它们设置为高电平即可使能对应的组。

实验演示

以 Arduino 为例，分别控制直流电机与步进电机

1. 直流电机

对于直流电机而言：M1EN、M2EN、M3EN、M4EN 接至了 Arduino 的 D6、D9、D10、D11，这些引脚具有 PWM 功能，因此控制 PWM 的占空比可以控制直流电机的转速。

analogWrite(uint8_t pin, int value)为写模拟引脚函数，对于参数 value 而言当设置为 0 时，输出低电平，设置为 255 时输出占空比为 100%的高电平。

```
int motor1_dir1 = 12;
int motor1_dir2 = 13;
int motor1_pwm  = 11;

void setup()
{
    pinMode(motor1_dir1,OUTPUT);
    pinMode(motor1_dir2,OUTPUT);
    pinMode(motor1_pwm,OUTPUT);

    digitalWrite(motor1_dir1,0);
    digitalWrite(motor1_dir2,1);
    digitalWrite(motor1_pwm,1);

}
void loop()
{
    analogWrite(motor1_pwm, 128);
    delay(500);
}
```

以控制一个直流电机为例：设置管脚为输出状态，然后配置初始状态，D12 管脚设置为低电平，D13 设置为高电平（当然这两个管脚可以反过来，那么电机转动方向也将改变），D11 输出高电平使能 L293 驱动芯片。

在 loop()函数中设置 D11 的 PWM 占空比为 50%(value = 128 即 50%)，也就间接的控制了电机转动速度。

2. 步进电机

以控制一个 28BYJ-48 步进电机为例： 该电机是一个五线四相八拍电机，接线如图：

Motor Control Shield	28BYJ-48 电机
5V	红线
M1A	橙线
M1B	黄线
M2A	粉线
M2B	蓝线

因为使用的为 M1A、M1B、M2A、M2B 来模拟时序控制，因此需要控制 M1EN、M2EN 为高电平使能 L293 芯片。

查阅 28BYJ-48 相关资料可以发现如下这样的两个表格：

	1	2	3	4	5	6	7	8
P1-红	VCC	VCC	VCC	VCC	VCC	VCC	VCC	VCC
P2-橙	GND	GND						GND
P3-黄		GND	GND	GND				
P4-粉				GND	GND	GND		
P5-蓝						GND	GND	GND

供电电压	相数	相电阻Ω	步进角度	减速比	启动频率P.P.S	转矩g.cm	噪声dB	绝缘介电强度
5V	4	50±10%	5.625/64	1:64	≥550	≥300	≤35	600VAC

第一个图说明了电机的四相八拍是如何控制的：

四相分别是 A,B,C,D，若要使 A 相导通则需要控制橙线接地，即控制 M1A 低电平。

类似的控制其他几个引脚,如果把四个控制引脚看作成 4 位 bit 连续数据，那么用 8 个字节即可表示这组逻辑的控制(橙线为最低位)：

```
Char BeatCode[8] = { //Stepper motor eight eight beat code
    0x0E, 0x0C, 0x0D, 0x09, 0x0B, 0x03, 0x07, 0x06
};
```

第二个图说明了控制的频率：

可以看到步进电机的启动频率，步进电机在空载情况下能够正常启动的最高脉冲频率，如果脉冲频率高于该值，电机就不能正常启动。给出的参数是≥550，单位是 P.P.S，即每秒脉冲数，这里的意思就是说：电机保证在你每秒给出 550 个步进脉冲的情况下，可以正常启动。那么换算成单节拍持续时间就是 1s/550=1.8ms,那么每个节拍的间隔大于该值即可转动，不过延时越久，电机转动一步所用的时间将会变长。

得到了上述信息即可以转动步进电机，但无法精确控制，28BYJ-48 是减速电机，其内部有四级减速，减速比为 1：64，转子转 64 圈，最终输出轴才会转一圈，也就是需要 64*64=4096 个节拍输出轴才转过一圈，如果每步 2ms,那么 2ms*4096=8192ms，需要 8 秒多才能够转一圈。但是实际上硬件上的减速比不一定符合标称值，这里采取经验值：4076 个节拍，实际误差万分之 0.56。

核心程序如下：

```
void Motor_Trun(BYTE Motor_dev, unsigned long Angle)
{
    struct MOTOR sMotor;
    if(Motor_dev == MOTOR_DEV_1){
        sMotor = sMotor1;
    }else if(Motor_dev == MOTOR_DEV_2){
        sMotor = sMotor2;
    }else{
        DEBUG("not motor device \r\n");
    }

    BYTE Index = 0;
    unsigned long beats = (Angle * 4076) / 360 ; //Need to turn the beat

    for(beats = beats; beats > 0; beats--){
        Motor_Setbit(sMotor, BeatCode[Index]);
        Index++;
        if(Index % 8 == 0){
            Index = Index & 0x07; //Greater than 8 clear 0
            DEBUG("*****************\r\n");
        }
    }
    Motor_Setbit(sMotor, 0x0f);
}
```