Motor Control Shield USER MANUAL

OVERVIEW

This module is a motor control expansion board of Arduino/NUCLEO. Onboard dual H-bridge driver chip D, allows to drive 4 DC motors or 2 stepping motors at one time.

Arduino and STM32 examples are provided for users.

FEATURES

- **Could drive 4 DC motors or 2 stepping motors at one time**
- **Configurable motor power supply via onboard jumper**
  - when using 5V motor power supply: drives four 5V DC motors at one time
  - when using adjustable motor power supply: drives two 1.25V-6.45V DC motors at one time (9V external power supply is required)
- **Onboard dual H-bridge driver D x 2**
  - each D can drive 2 DC motors or 1 stepping motor at one time
  - totally 4 channel H-bridge, with 600mA output current (peak 1.2A) per single bridge
  - ESD protection

PIN CONFIGURATION

Motor Control Shield	Arduino	NUCLEO	DC Motor	Stepping Motor
3V	3V	3V	N/A	common terminal
M1A	D2	PA10	DC motor 1	stepping motor 1
M1B	D3	PB3		

M2A	D4	PB5	DC motor 2	
M2B	D5	PB4		
M3A	D7	PA8	DC motor 3	stepping motor 2
M3B	D8	PA9		
M4A	D12	PA6	DC motor 4	
M4B	D13	PA5		
M1EN	D6	PB10	M1A, M1B Enable	
M2EN	D9	PC7	M2A, M2B Enable	
M3EN	D10	PB6	M3A, M3B Enable	
M4EN	D11	PA7	M4A, M4B Enable	

Note: The M1EN, M2EN, M3EN and M4EN have not silk screen printing on board, they are hardware pins of L293 driver chips. You can enable them by setting them to HIGH

DEMO CODE

Use Arduino as example, shows how to control DC motor and stepping motor.

1. DC motor

To control DC motor, you need to connect M1EN, M2EN, M3EN and M4EN to D6, D9, D10, D11 of Arduino. These are the PWM pin of Arduino, so we could control the speed of DC motor by controlling the Duty ratio of PWM.

analogWrite(uint8_t pin, int value) used for writing analog pins. If set the parameter value to 0, the pin will output LOW level, and if it is set as 255, the pins will output HIGH level which duty ratio is 100%.

```
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);
}
```

To control a DC motor:

Configure pin to output status firstly, then configure the initial status. Set D12 to LOW, D13 to HIGH (For changing the rotation direction, you can reverse the level of these pins). D11 is set to HIGH to enable the L293 driver chip.

In the loop() function, the PWM duty ratio of D11 is set to 50% (value = 128 is equal to 50%). This value could control the speed of motor.

2. Stepping motor

For example, control a 28BYJ-48 stepping motor: this motor is a 5-wires 4-phase motors. Connect it as below:

Motor Control Shield	28BYJ-48
5V	Red
M1A	Orange
M1B	Yellow
M2A	Pink
M2B	Blue

Because it uses M1A, M1B, M2A and M2B to do the timing control, here we need to set M1EN, M2EN to high to enable the L293

Refer to documents of 28BYJ-48 we could get:

	1	2	3	4	5	6	7	8
P1-Red	VCC	VCC	VCC	VCC	VCC	VCC	VCC	VCC
P2-Orange	GND	GND						GND
P3-Yellow		GND	GND	GND				
P4-Pink				GND	GND	GND		
P5-Blue						GND	GND	GND

Voltage	Phase	Phase resistor Ω	Stepped-Angle	Reduction-gear ratio	Start frequency	Torque	Noise	Dielectric Strength
5V	4	50±10%	5.625/64	1:64	⩾550	⩾300	⩽35	600VAC

The first figure shows how to control the control the 4-phase and 8-steps of motor:

The 4 phases are A, B, C, D. If you want to make the A-phase on-state, you need to connect orange wire to GND, that is set M1A to Low.

Control the other pins with the same way. If take the 4 Pins as 4-bits continuous data, we could use 8 bytes to control them: (orange pin is the lowest byte)

Char BeatCode[8] = { //Stepper motor eight eight beat code

0x0E, 0x0C, 0x0D, 0x09, 0x0B, 0x03, 0x07, 0x06

};

The second figure shows the frequency parameters of control:

There is start frequency, it explains the maximum pulse rate of stepping motor while no-load starting. If the pulse rate is higher that this value, the motor could not work. The value is ⩾ 550, Its uint is P.P.S means pulses per second. It is that motor could start properly if you input 550 pulses every second. And we could get that, every pulse cause 1s/550=1.8ms. So if the time interval longer than 1.8ms, the motor could work. Of cause the longer the interval, the longer motor rotates one step.

With the information above, we could rotate the motor, but still cannot control it precisely. 28BYJ-48 is a geared motor. The reduction-gear ratio is 1:64. The output shaft rotate a round

when the rotator turns 64 rounds, it is that 64*64=4096 pulses are required for a round, If every pulse cost 2ms, it requires 2ms*4096=8192ms, about 8s. In fact, the real ratio generally isn't equal to standard value. Here we use value 4076 pluses after testing, the tolerance is about 0.56‰. The main code is that:

```
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);
}
```