BH-MSD4805 步进电机驱动器
用户手册

修订历史

日期	版本	更新内容
2016/11/26	1.0.0	

文档说明

本手册旨在帮助用户正确构建 BH-MSD4805 步进电机驱动器的使用环境，引导用户快速使用该驱动器。

目录

1. BH-MSD-4805 简介

1.1 BH-MSD4805 简介

BH-MSD4805 是秉火科技推出的一款智能步进电机驱动器。它是一款以双极恒流 PWM 驱动输出控制电机的驱动器，驱动电压范围 DC12V～48V，适合外径为 42mm、57mm、86mm 系列，驱动电流在 5A 以下的所有两相混合式步进电机。根据驱动器提供的 8 位拨码开关可以轻松的实现对不同电机电流及不同细分步数的精确控制。带有自动半流技术，可以大大降低电机的功耗及发热量，输入信号都经过光耦隔离，具有很强的抗干扰能力，能适应恶劣的工作环境。

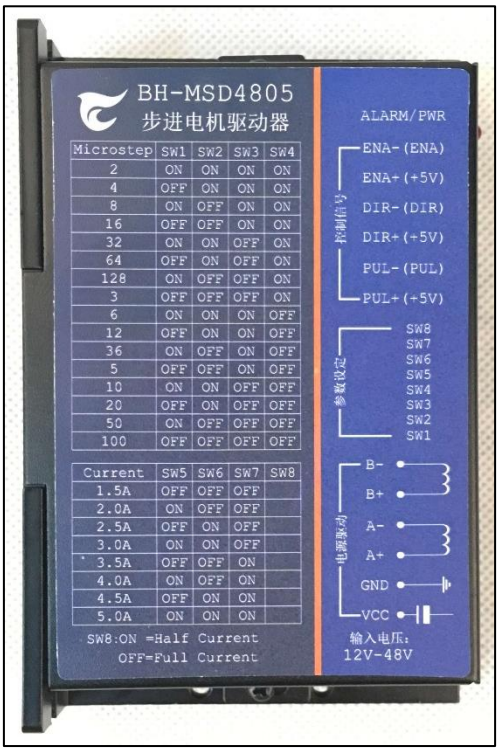

图 1-1 MSD4805 外观图

1.2 特性参数

参数	说明
额定电压	直流：12V ~ 48V
额定电流	0.75A ~ 5.0A
驱动方式	双极恒流 PWM 驱动输出
工作温度	0℃ ~ 80℃
结构尺寸	118(mm)*75.5(mm)*33(mm)
应用领域	数控设备、雕刻机等设备

2. 硬件测试

本驱动器配套 STM32 驱动程序，可直接使用秉火指南者、霸道及挑战者开发板进行测试。按要求使用杜邦线把模块连接到开发板，并下载程序即可。

2.1 硬件连接

BH-MSD4805 驱动器外观见图 2-1，驱动器右侧分别是电源及故障指示灯、控制信号接口、参数设定拨码开关、电源驱动接口，在其端子的正上方是对应引脚名称的丝印。

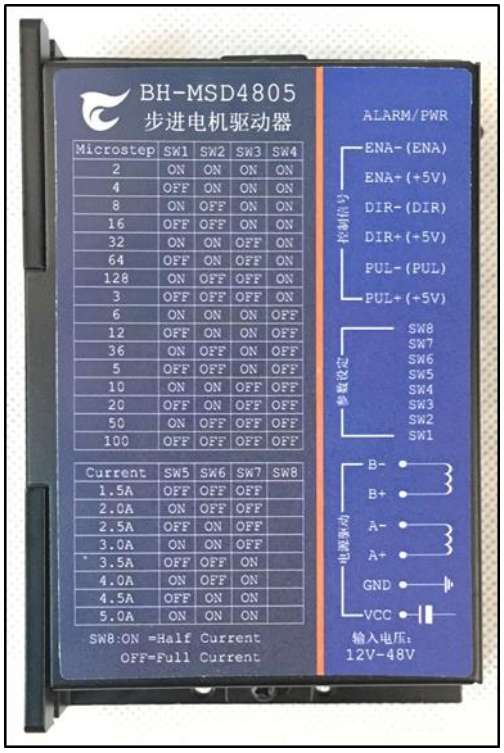

图 2-1 BH-MSD4805 驱动器外观

表 2-1 BH-MSD4805 驱动器控制信号说明

序号	引脚名称	引脚定义	功能说明
1	ENA-(ENA)	输出使能负端	控制器的输出是通过该组信号使能，又称脱机信号。当此信号有效时，输出关闭，电机绕组电流为零，电机为无力矩状态，可以自由转动电机，适合需要手动调整电机的场合。
2	ENA+(+5V)	输出使能正端	
3	DIR-(DIR)	方向控制负端	电机的方向控制信号，当此信号有效时，电机顺时针转动，当此信号无效时，电机逆时针旋转。
4	DIR+(+5V)	方向控制正端	
5	PUL-(PUL)	脉冲控制负端	电机的转动控制信号，驱动器接收到的脉冲信号电机就会按照既定的方向旋转。电机的角位移与脉冲的数量成正比，速度与脉冲的频率成正比。通常脉冲的有效宽度>=5us,频率<=125KHz。
6	PUL+(+5V)	脉冲控制正端	

表 2-2 BH-MSD4805 驱动器拨码开关说明

序号	引脚名称	引脚定义	功能说明
1	SW1	细分设定	由 SW1—SW4 四个拨码开关来设定驱动器微步细分数，其有 16 档微步细分设定。用户设定微步细分时需要重新上电微步细分才生效，具体输出微步细分的设定，见驱动器的细分设置由拨码开关的 SW1~SW4 来设定，默认为 100 细分，一般的两相四线制步进电机的步进角都是 1.8°，因此电机旋转一圈需要 360°/1.8°=200 个脉冲，这里 100 细分转一圈需要的脉冲数为 200*100=20000 个。详细设定见表 2-7。 表 2-7。
2	SW2		
3	SW3		
4	SW4		
5	SW5	电流设定	由 SW5—SW8 四个拨码开关来设定驱动器输出电流，其输出电流共有 14 档。其中 SW8 为全局电流设定开关，如果设定为 ON,则输出结果为设定值的一半。具体输出电流的设定，见驱动器的电流设置由拨码开关的 SW5~SW8 来设定，默认为 1.5A。这个电流值需要根据步进电机的额定电流来设定。一般建议驱动器的输出电流设定和电机额定电流差不多或者小一点，详细设定见表 2-7。 表 2-8 电流参数使用拨码开关设置说明。
6	SW6		
7	SW7		
8	SW8		

表 2-3 BH-MSD4805 驱动器电源驱动端子说明

序号	引脚名称	引脚定义	功能说明
1	A-	A 相绕组负端	驱动器的 A 相绕组驱动输出，跟电机的 A 相绕组相连
2	A+	A 相绕组正端	
3	B-	B 相绕组负端	驱动器的 B 相绕组驱动输出，跟电机的 B 相绕组相连
4	B+	B 相绕组正端	
5	GND	电源负端	驱动器的电压供电口，驱动电压应该满足直流：12V ~ 48V
6	VCC	电源正端	

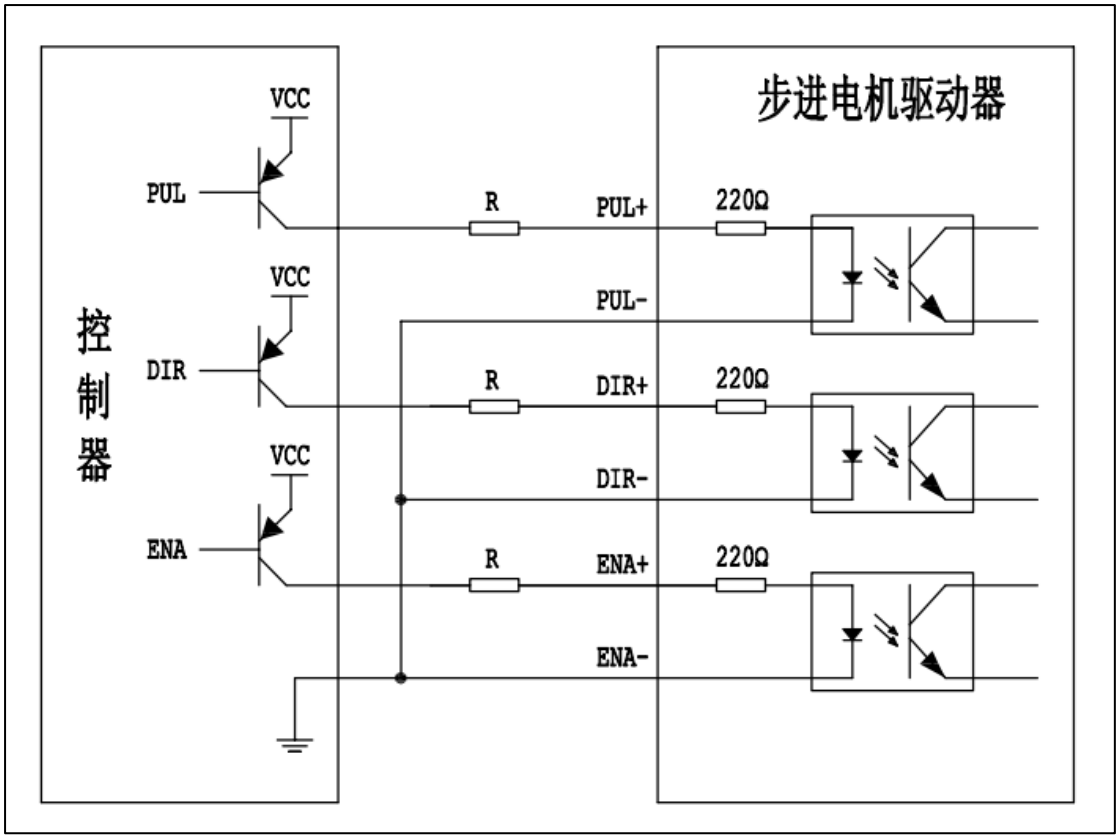

图 2-2 BH-MSD4805 驱动器与控制器共阴方式接线图

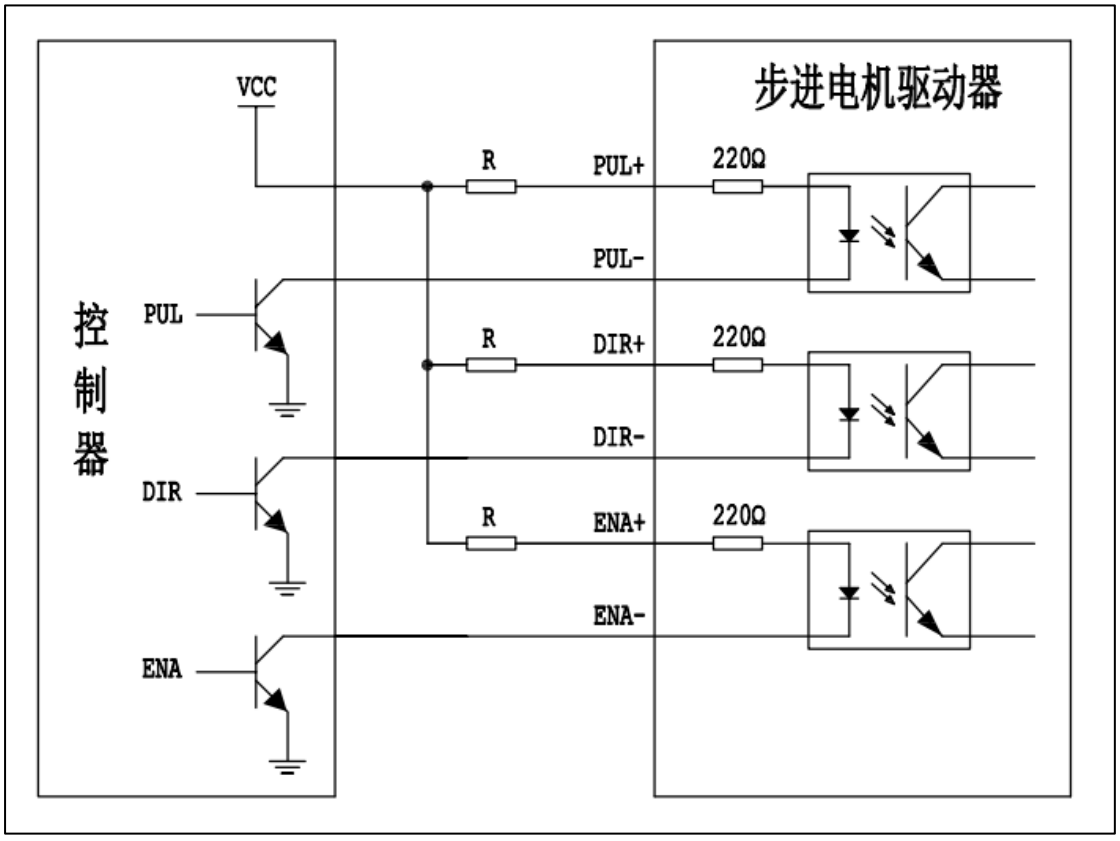

图 2-3 BH-MSD4805 驱动器与控制器共阳方式接线图

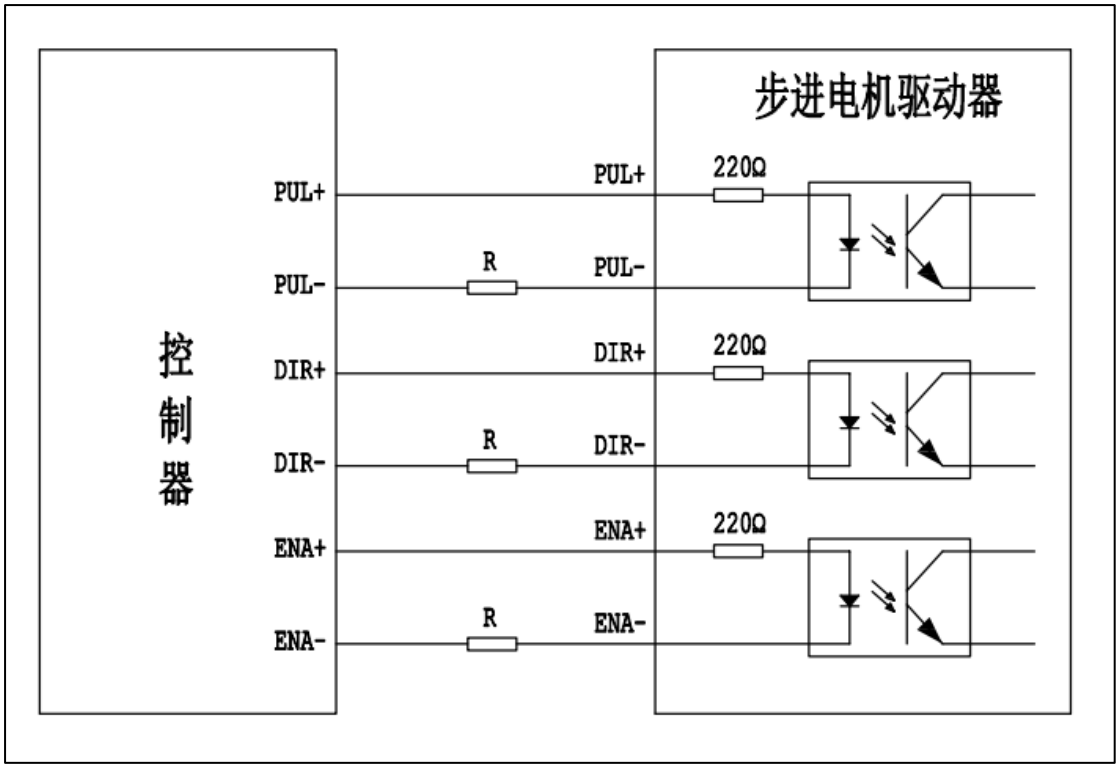

图 2-4 BH-MSD4805 驱动器与控制器差分方式接线图

当输入信号高于 5V 时，可根据需要外接限流电阻。限流电阻 R 值选取如表 2-4 所示：

表 2-4 限流电阻 R 值说明

控制信号电平(V)	+5V	+12V	+24V
限流电阻 R(Ω)	R=0	R=1K(1/4W)	R=2K(1/4W)

BH-MSD4805 驱动器引脚的具体说明见表 2-1 和表 2-3，本说明采用的是图 2-2 所示的共阴方式进行开发板和驱动器之间的连接，详细接线见下表表 2-5。驱动器与步进电机连接见表 2-6。驱动器与指南者连接效果见图 2-5。

表 2-5 驱动器与各个开发板接线说明

驱动器引脚	F103-指南者	F103-霸道	F429-挑战者
ENA-	GND	GND	GND
ENA+(5V)	PC4	PC4	PB14
DIR-	GND	GND	GND
DIR+(5V)	PB14	PB14	PC0
PUL-	GND	GND	GND
PUL+(5V)	PA3	PA3	PA3

表 2-6 驱动器与电机接线说明

驱动器引脚	电机绕组接线
A+	蓝色
A-	红色
B+	绿色
B-	黑色

注：电机驱动器的 VCC、GND 之间输入 12V~48V 的直流电源

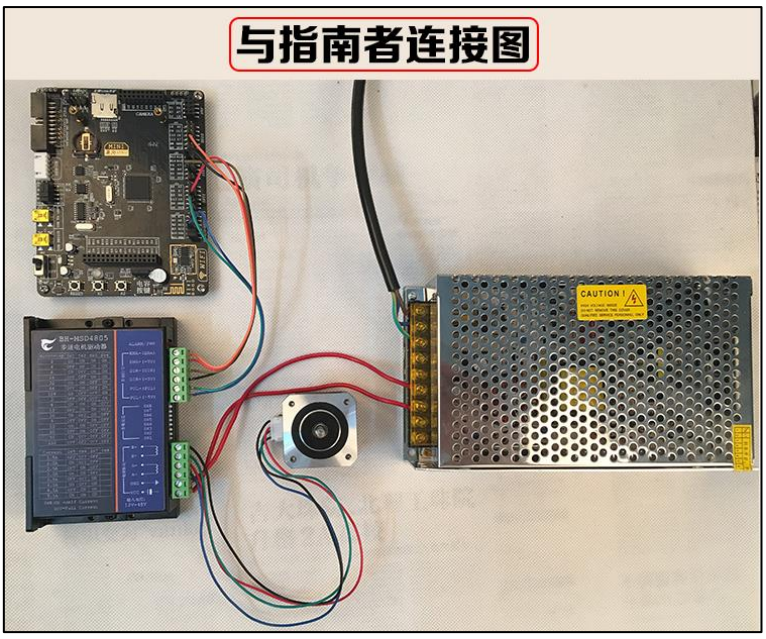

图 2-5 驱动器与指南者开发板连接图

2.2 测试流程

2.2.1 参数设置

1. 细分微步参数设置

驱动器的细分设置由拨码开关的 SW1~SW4 来设定，默认为 100 细分，一般的两相四线制步进电机的步进角都是 1.8°，因此电机旋转一圈需要 360°/1.8°=200 个脉冲，这里 100 细分转一圈需要的脉冲数为 200*100=20000 个。详细设定见表 2-7。

表 2-7 细分微步参数使用拨码开关设置说明

细分微步	脉冲	SW1	SW2	SW3	SW4
2	400	ON	ON	ON	ON
4	800	OFF	ON	ON	ON
8	1600	ON	OFF	ON	ON
16	3200	OFF	OFF	ON	ON
32	6400	ON	ON	OFF	ON
64	12800	OFF	ON	OFF	ON
128	25600	ON	OFF	OFF	ON
3	600	OFF	OFF	OFF	ON
6	1200	ON	ON	ON	OFF
12	2400	OFF	ON	ON	OFF
36	7200	ON	OFF	ON	OFF

5	1000	OFF	OFF	ON	OFF
10	2000	ON	ON	OFF	OFF
20	4000	OFF	ON	OFF	OFF
50	10000	ON	OFF	OFF	OFF
100	20000	OFF	OFF	OFF	OFF

2. 电流参数设置

驱动器的电流设置由拨码开关的 SW5~SW8 来设定，默认为 1.5A。这个电流值需要根据步进电机的额定电流来设定。一般建议驱动器的输出电流设定和电机额定电流差不多或者小一点，详细设定见表 2-7。

表 2-8 电流参数使用拨码开关设置说明

电流	SW5	SW6	SW7	SW8
0.75A	OFF	OFF	OFF	ON
1.00A	ON	OFF	OFF	ON
1.25A	OFF	ON	OFF	ON
1.50A	OFF	OFF	OFF	OFF
1.75A	OFF	OFF	ON	ON
2.0A	ON	OFF	OFF	OFF
2.25A	OFF	ON	ON	ON
2.50A	OFF	ON	OFF	OFF
3.00A	ON	ON	OFF	OFF
3.50A	OFF	OFF	ON	OFF
4.00A	ON	OFF	ON	OFF
4.00A	OFF	ON	ON	OFF
5.00A	ON	ON	ON	OFF

3. 自动半流功能

步进电机在工作时，若切断脉冲输入，电机即处于刹车状态，这是步进电机的优点，但此时电机通过的电流最大，噪声也大，电机发热量剧增，会影响电机性能及使用寿命。为了解决这个问题，我们设计的步进电机驱动器带有自动半流功能，当切断脉冲输入 0.52s 后驱动器自动进入自动半流模式，电机电流值会变为工作时的一半，电机的发热量会减小，噪声也会降低。

2.2.2 控制步进电机加减速运动

1. 准备测试环境

硬件平台根据前面的硬件连接和参数设置的说明已经准备好，这里开始我们来把软件平台准备好。找到配套资料里的例程“1. BH-MSD4805 步进电机驱动（指南者）”，使用 MDK 编译并下载该程序到指南者开发板，复位开发板让程序运行。PC 端把秉火多功能调试助手串口终端打开，并连接上开发板对应的串口。

工程所在目录：“开发板配套例程\xx”。

2. 正常运行的实验现象

我们提供的例程，能够实现步进电机的正反转、加减速、脱机模式。MDK 编译并下载该程序到指南者开发板，复位开发板让程序运行。使用串口终端来控制步进电机，程序运行后会打印如下帮助菜单，见图 2-6 串口终端打印信息；

图 2-6 串口终端打印信息

下面我们详细介绍串口控制命令，见表 2-9。

表 2-9 串口命令说明

串口指令	参数取值范围	用法（注意空格）末尾以回车键结束	用法含义
?	-	?	打印帮助菜单
a[data]	71~32000(单位：0.01rad/s²)	a 32000	加速度为 320rad/s²
d[data]	71~32000(单位：0.01rad/s²)	d 10000	减速度为 320rad/s²
s[data]	12~3000(单位：0.01rad/s)	s 3000	最大速度为 30rad/s
m[data]	-2147483647~2147483647(步)	m 20000	移动步数 20000
Move [step] [accel] [decel] [speed]	范围同上	move 20000 5000 5000 3000	以加速度 50 rad/s²、减速度 50rad/s²、最大速度 30 rad/s，移动 20000 步
Enter 键	-	Enter 键	重复最后一次运动

- 默认情况下，我们只需在串口终端输入栏下按一下回车键然后发送，开发板收到指令后，会控制步进电机以 320 rad/s² 的加速度、320 rad/s² 的减速度、最大速度 30 rad/s，移动 40000 步即旋转四圈。我们可以直观的看到电机由加速到匀速然后到减速的过程。到达终点后串口会打印当前步进电机的位置及相关的参数见图 2-7。

图 2-7 串口终端发送回车键打印信息

- a:设置加速度，例如我们需要设置加速度为 100rad/s²，加速度的单位为 0.01rad/s²，经过计算 a 的值应该为 100/0.01=10000，所以在串口中断输入栏下输入“a 10000” +回车键然后按发送，开发板收到指令后，串口会打印对应的加速度的值，见图 2-8。

图 2-8 串口终端设置加速度打印信息

- d:设置减速度，例如我们需要设置减速度为 100rad/s²，加速度的单位为 0.01rad/s²，经过计算 d 的值应该为 100/0.01=10000，所以在串口中断输入栏下输入“d 10000”+回车键然后按发送，开发板收到指令后，串口会打印对应的减速度的值，见图 2-9。

图 2-9 串口终端设置减速度打印信息

- s:设置最大速度，例如我们需要设置最大速度为 30rad/s，加速度的单位为 0.01rad/s，经过计算 s 的值应该为 30/0.01=3000，所以在串口中断输入栏下输入“s 3000”+回车键然后按发送，开发板收到指令后，串口会打印对应的最大速度的值，见图 2-10。

图 2-10 串口终端设置最大速度打印信息

- m:移动固定步数，步数如果是正数就顺时针旋转，如果是负数就逆时针旋转，例如我们需要移动 10000 步，在串口中断输入栏下输入“m 10000”+回车键然后按发送，开发板收到指令后，串口会打印对应的加减速度，最大速度、移动步数及电机当前位置，见图 2-11。

图 2-11 串口终端打印移动固定步数信息

- move:移动固定步数同时设置参数，步数如果是正数就顺时针旋转，如果是负数就逆时针旋转，同时也一起设置加减速度，最大速度。例如我们需要以 320 rad/s² 的加速度、320 rad/s² 的减速度、最大速度 10 rad/s 移动 10000 步，在串口中断输入栏下输入“move 10000 32000 32000 1000”+回车键然后按发送，

开发板收到指令后，串口会打印对应的加减速度，最大速度、移动步数及电机当前位置，见图 2-12。

图 2-12 串口终端打印移动固定步数及参数信息

- t:以程序预置的方式运行，在串口中断输入栏下输入“t”+回车键然后按发送，开发板收到指令后，串口会打印对应的加减速度，最大速度、移动步数及电机当前位置，见图 2-13。

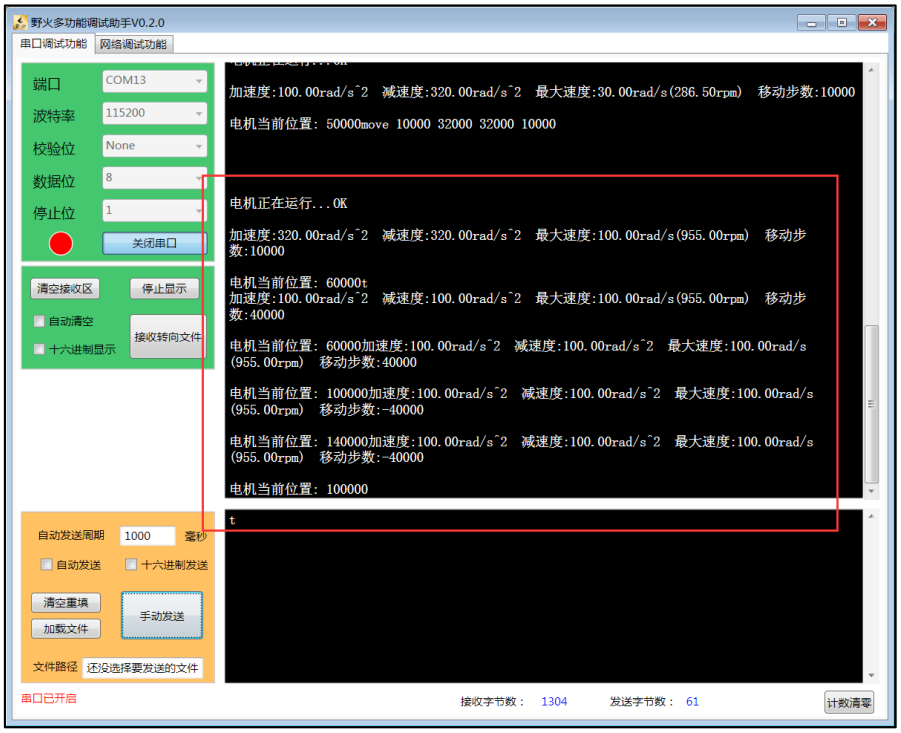

图 2-13 串口终端打印测试步进电机运动信息

- KEY1，在电机上电工作的时候，如果我们需要手动旋转步进电机，只需要步进电机停止转动后，我们按下按键 KEY1，进入脱机状态，此时我们可以手动去调整步进电机，串口打印信息见图 2-14。

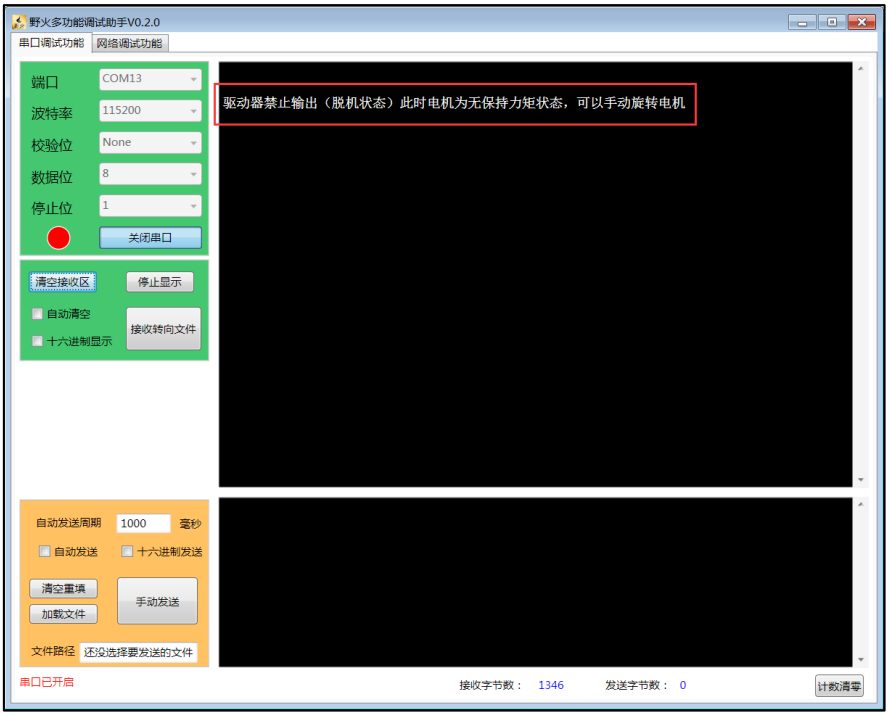

图 2-14 串口终端打印脱机状态信息

- KEY2，在脱机状态的时候，如果我们恢复步进电机正常工作状态，只需要步进电机停止转动后，我们按下按键 KEY2，进入正常工作状态，此时我们可以串口指令去控制步进电机，串口打印信息见图 2-15 图 2-12。

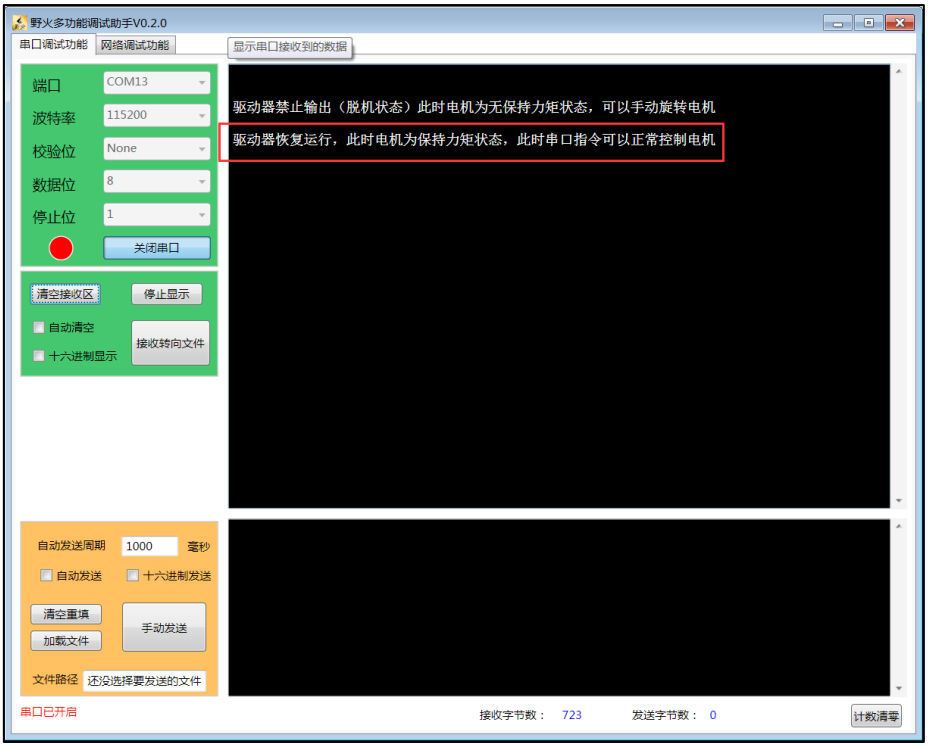

图 2-15 串口终端打印恢复正常工作状态信息

3. 指示灯状态指示

驱动器上有两颗 LED 指示灯，PWR 是电源指示灯，正常工作时常亮；ALARM 是状态指示灯，可以指示欠压状态，短路过流状态，操作错误状态。具体见表 2-10。

表 2-10 LED 指示灯状态说明

LED	LED 状态	状态说明
PWR	绿色常亮	驱动器电源工作正常
	不亮	驱动器电源不正常
ALARM	红色闪烁	欠压指示，电源电压低于 8V 时
	红色闪烁	过流指示，输出短路造成过流
	红色闪烁	系统运行过程中操作拨码开关
	不亮	系统无错误

3. 注意事项及常见问题

1. 拨码开关设置

为了避免操作失误，造成系统故障，请先设置好拨码开关再上电，如果需要重新设置请断电后重新设置，然后上电即可生效。

2. 常见问题及处理方法

常见现象	可能问题	解决方法
电机不转	电源灯不亮	检查电源供电电路，接线极性
	使能信号为高	使能信号置低
	脉冲频率过高	降低脉冲频率
	电机接线错误	检查电机 AB 相绕组的接线
电子堵转	脉冲频率过高	降低脉冲频率
	功率不够	增大电源功率
报警指示灯闪烁	电源欠压	电源正常工作范围 12V~48V
	输出短路	检查输出线路
	设置拨码开关	恢复原来的状态
电机转向错误	电机接线错误	交换 A+和 A-或者 B+和 B-接线
	控制方向信号出错	检查控制信号连接
位置不准确	细分设置错误	拨码开关设置细分要跟程序一致
	脉冲频率过高	降低脉冲频率

4. 配套程序说明

BH-MSD4805 驱动器一共配套了 3 个例程，用户可根据需求选择相应的程序来学习。演示程序的算法参考了《AVR446_Linear speed control of stepper motor.pdf》。

例程所在目录：“开发板配套例程”。

程序	说明
1. BH-MSD4805 测试程序 F103(指南者)	带有速度控制，可以实现加减速控制，精确移动到目标位置；脱机模式控制，手动调整电机
2. BH-MSD4805 测试程序 F103(霸道)	带有速度控制，可以实现加减速控制，精确移动到目标位置；脱机模式控制，手动调整电机
3. BH-MSD4805 测试程序 F429(挑战者)	带有速度控制，可以实现加减速控制，精确移动到目标位置；脱机模式控制，手动调整电机

4.1 驱动原理

1. 步进电机

步进电机是一种将电脉冲转化为角位移的执行机构。通俗一点讲：当步进驱动器接收到一个脉冲信号，它就驱动步进电机按设定的方向转动一个固定的角度（或步进角）。可以通过控制脉冲个数来控制角位移量，从而达到准确定位的目的；同时也可以通过控制脉冲频率来控制电机转动的速度和加速度，从而达调速的目的。因此在需要准确定位或调速控制时均可考虑使用步进电机。

2. 细分驱动

细分驱动模式具有低速振动极小和定位精度高两大优点。对于有时需要低速运行(即电机转轴有时工作在 60rpm 以下)或定位精度要求小于 0.9° 的步进应用中，细分型步进电机驱动器获得广泛应用。其基本原理是对电机的两个线圈分别按正弦和余弦形的台阶进行精密电流控制，从而使得一个步距角的距离分成若干个细分步完成。例如 100 细分的驱动方式可使每圈 200 标准步的步进电机达到每圈 200*100=20000 步的运行精度(即 0.018°)。

3. 速度属性

步进电机有个缺陷是速度提高后力矩会缺失，即速度越快力矩越小。因为当步进电机转动时，电机各相绕组的电感将形成一个反向电动势；速度（或脉冲频率）越高，反向电动势越大。在反向电动势的作用下，电机的相电流随速度（或脉冲频率）的增大而减小，从而导致力矩下降。当速度（或脉冲频率）增加到一定值时，步进电机已无法带动任何负载，而且只要受到很小的扰动，就会振荡、失步，甚至停转。出现共振速度也是力矩降低的另外一个原因。力矩与速度关系见图 4-1。

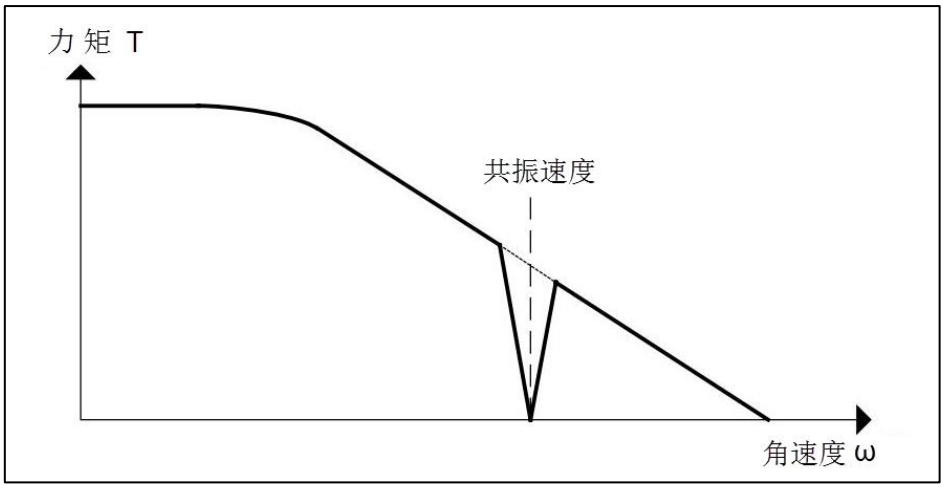

图 4-1 力矩与角速度关系

4. 步进电机基础方程

若要步进电机运动起来必须在电机各个绕组上以正确的顺序施加电流。这部分工作已经由驱动器去完成，我们只需要在驱动器施加脉冲和正确的方向即可让电机运动起来。如果要步进电机以恒定的速度旋转，我们就需要以固定的频率发送脉冲，我们通过单片机的定时器来实现脉冲的发送，如图 4-2 所示，t_0为脉冲发送的起始时刻，t_1为发送第二个脉冲的时刻，t_2为发送第三个脉冲的时刻。t_0与t_1之间的时间间隔为$\delta_t = c_0 t_t$，其中c_0为定时器在t_0与t_1这段时间的的计数值，t_t为定时器的计数周期。t_1与t_2之前的时间间隔为$\delta_t = c_1 t_t$，其中c_1为定时器在t_1与t_2这段时间的计数值，t_t为定时器的计数周期。

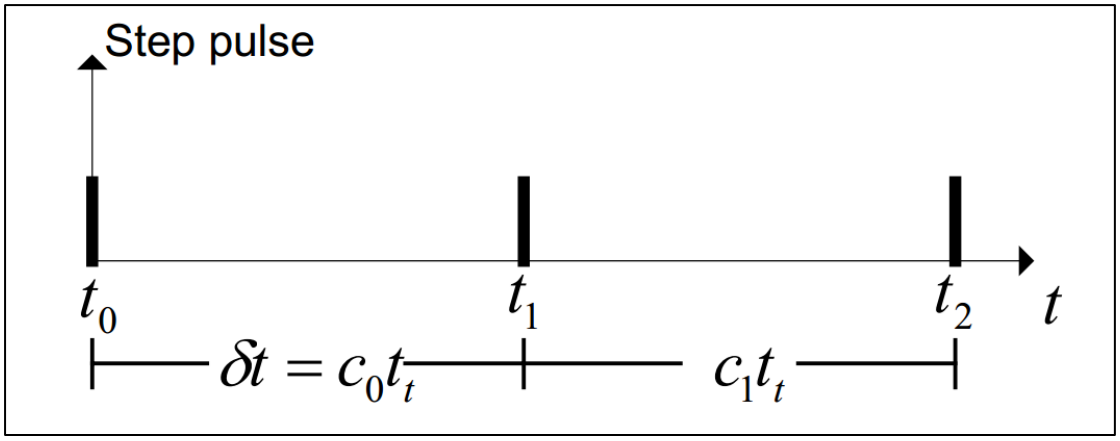

图 4-2 步进电机脉冲

我们可以假定产生脉冲的定时器的计数频率为f_t，那么$t_t = \frac{1}{f_t}$,可以推出以下公式

$$\delta_t = c_0 t_t = \frac{c_0}{f_t}(s) \qquad (4—1)$$

通过给出了电机步距角 α、位置 θ 和速度 ω，我们可以得到

$$\alpha = \frac{2\pi}{spr}(rad) \qquad (4—2)$$

$$\theta = n\alpha\ (rad) \qquad (4—3)$$

$$\omega = \frac{\theta}{\delta_t}\ (rad/s) \qquad (4—4)$$

其中 spr 为英文 steps per round 的缩写，这里用来作为步进电机旋转一圈的脉冲数，n 为脉冲数，1 $(rad/s) \approx 9.55\ (rpm)$，rpm 为英文 rounds per minutes 的缩写，即圈速。

5. 线性速度坡道

为了在电机启动过程中更平滑，我们就需要引入加速度和减速度。如图 4-3 所示。

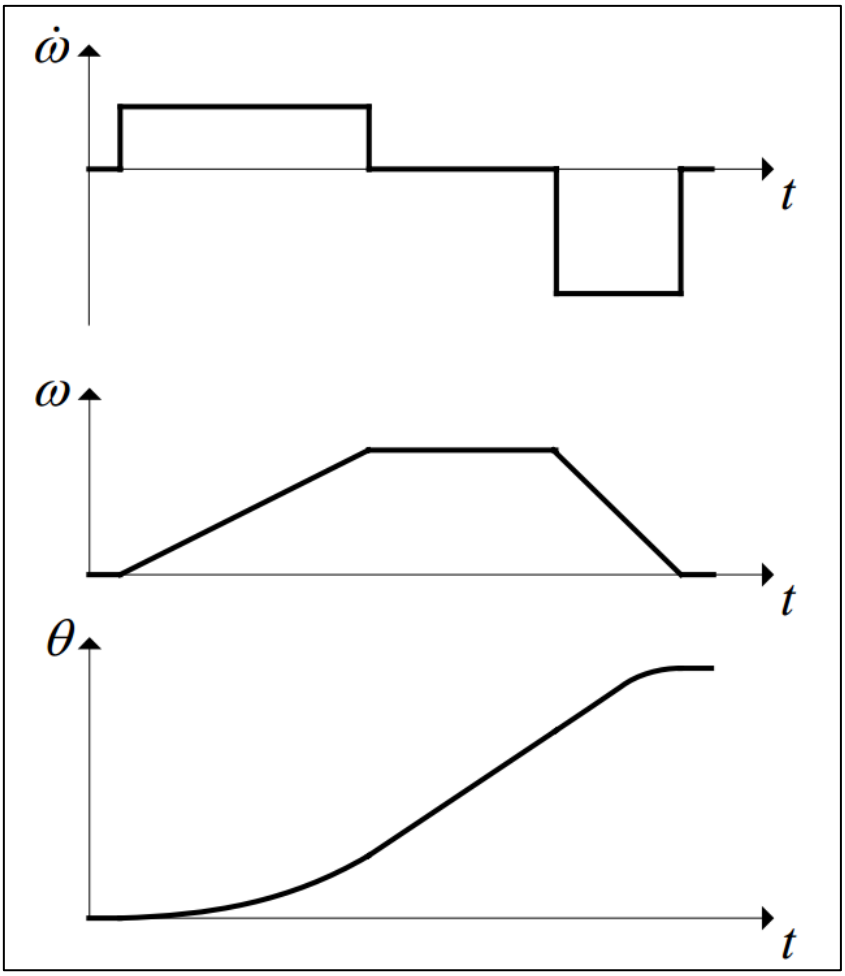

图 4-3 加速度，速度，位移的相互关系

步进电机的速度由脉冲之间的时间延迟 δ_t 控制。为了使步进电机的速度更加接近速度坡道，这些时间都需要通过计算得到。使用定时器的计数频率来离散步进控制步进电机运动和处理延时。

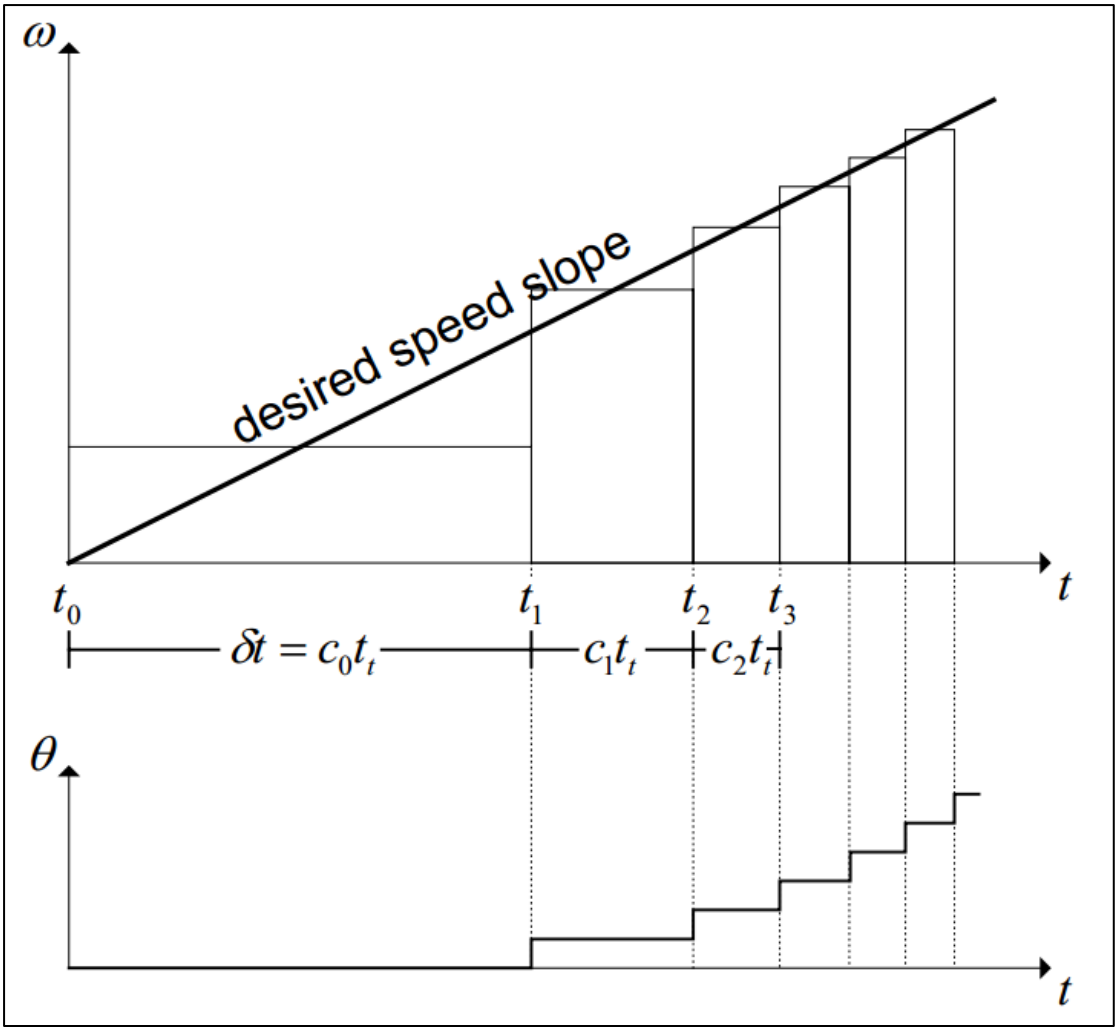

图 4-4 角位移与速度（脉冲）关系

如图 4-4 所示，期望的线性速度坡度是通过定时器的拟合，无限接近。

6. 精确计算步进一步的延时

某个时刻的速度可以由加速度求得

$$\omega(t) = \int_{\tau=0}^{t} \dot{\omega}\, d\tau = \dot{\omega}t \qquad (4—5)$$

某个时刻的位移可以由加速度求得

$$\theta(t) = \int_{\tau=0}^{t} \omega(\tau) d\tau = \frac{1}{2}\dot{\omega}t^2 \qquad (4—6)$$

第 n 步的脉冲产生的轴交角θ = nα，联合式（4—3）、（4—5）和（4—6）可以推出

$$t_n = \sqrt{\frac{2n\alpha}{\dot{\omega}}} \qquad (4—7)$$

两步之间的延时为

$$c_n t_t = t_{n+1} - t_n = \sqrt{\frac{2\alpha}{\dot{\omega}}}\left(\sqrt{n+1} - \sqrt{n}\right) \qquad (4—8)$$

最终可以求得定时器的计数值

$$c_n = \frac{1}{t_t}\sqrt{\frac{2\alpha}{\dot{\omega}}}\left(\sqrt{n+1}-\sqrt{n}\right) \quad (4—9)$$

由式（4—9）可以得到第一个脉冲定时器计数值c_0以及第 n 个脉冲定时器计数值c_n

$$c_0 = \frac{1}{t_t}\sqrt{\frac{2\alpha}{\dot{\omega}}} \quad (4—10)$$

$$c_n = c_0\left(\sqrt{n+1}-\sqrt{n}\right) \quad (4—11)$$

由于 MCU 的计算能力是有限的，连续两次计算开方根会很费时，因此我们必须考虑使用多项式来展开式（4—11）来减少运算。式（4—12）为泰勒公式的一个特例麦克劳林公式。

$$\sqrt{1\pm\frac{1}{n}} = 1\pm\frac{1}{2n}-\frac{1}{8n^2}+O\left(\frac{1}{n^3}\right) \quad (4—12)$$

第 n 个脉冲定时器计数器值c_n可以使用式（4—12）进行化简。

$$\frac{c_n}{c_{n-1}} = \frac{c_0\left(\sqrt{n+1}-\sqrt{n}\right)}{c_0\left(\sqrt{n}-\sqrt{n-1}\right)} = \frac{\sqrt{n+1}-\sqrt{n}}{\sqrt{n}-\sqrt{n-1}} = \frac{\frac{\sqrt{n+1}-\sqrt{n}}{\sqrt{n}}}{\frac{\sqrt{n}-\sqrt{n-1}}{\sqrt{n}}} = \frac{\sqrt{1+\frac{1}{n}}-1}{1-\sqrt{1-\frac{1}{n}}}$$

$$= \frac{\left(1+\frac{1}{2n}-\frac{1}{8n^2}+O\left(\frac{1}{n^3}\right)\right)-1}{1-\left(1-\frac{1}{2n}-\frac{1}{8n^2}+O\left(\frac{1}{n^3}\right)\right)} = \frac{\frac{1}{2n}-\frac{1}{8n^2}+O\left(\frac{1}{n^3}\right)}{\frac{1}{2n}+\frac{1}{8n^2}-O\left(\frac{1}{n^3}\right)} \approx \frac{4n-1}{4n+1}$$

（4—13）

由式（4—13）最后得到c_n

$$c_n = c_{n-1}\left(\frac{4n-1}{4n+1}\right) = c_{n-1}\frac{(4n+1)-2}{4n+1} = c_{n-1}-\frac{2c_{n-1}}{4n+1} \quad (4—14)$$

这个公式比连续开两次方的计算方式快很多，但是代入原式时发现有偏差，我们可以将 c_0乘一个系数 0.676 来解决这个误差。

7. 加速度的变化

根据上面公式可以知道，加速度与c_0和 n 相关。如果需要改变加速度或者减速度，那么就要重新计算一个 n 值。时间t_n和 n 作为加速度、速度和步进角的参数，我们可以得到

$$\mathrm{t}_n = \frac{\omega_n}{\dot{\omega}} \quad (4—15)$$

$$\mathrm{n} = \frac{\dot{\omega}t_n^2}{2\alpha} \quad (4—16)$$

联合式（4—15）和（4—16）得到

$$\mathrm{n}\dot{\omega} = \frac{\omega_n^2}{2\alpha} \quad (4—17)$$

式（4—17）表示达到给定的最大速度时需要的步数与加速度成反比，由于电机加速到最大时跟电机开始减速时的速度是一样的，我们可以得到

$$n_1\dot{\omega}_1 = n_2\dot{\omega}_2 \qquad (4—18)$$

这样我们只需要改变 n 的值就可以改变加速度的值。见图 4-5。

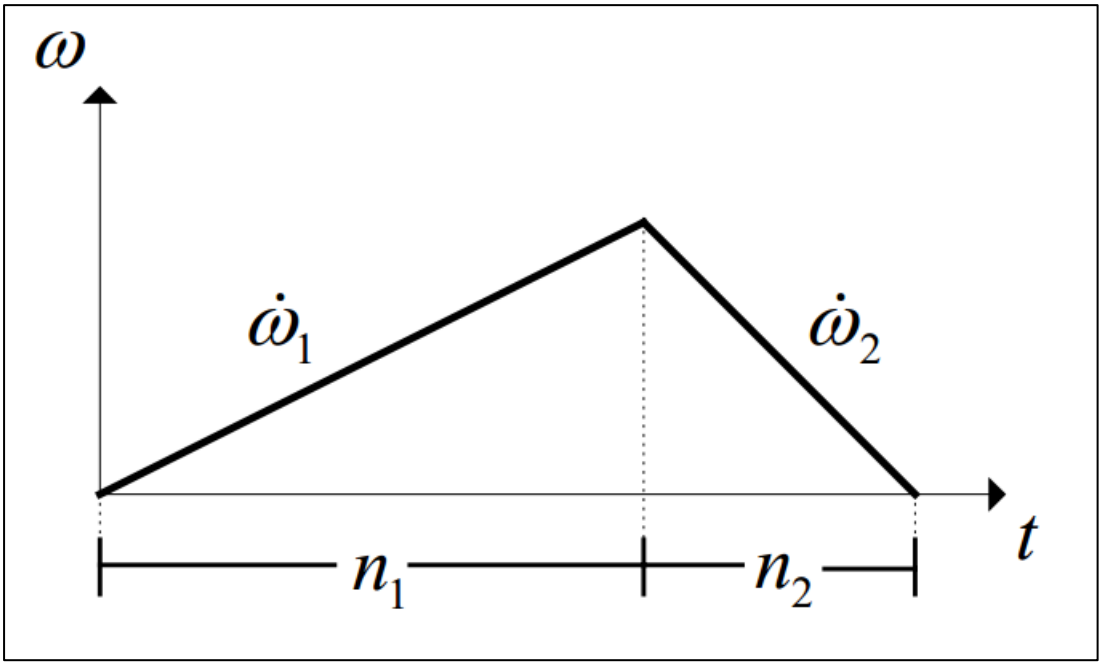

图 4-5 加减速坡度

移动给定的步数，必须在适当的步数时开始减速，使其结束的时候速度为 0。由式（4—18）可以得到n_1

$$n_1 = \frac{(n_2+n_1)\dot{\omega}_2}{(\dot{\omega}_1+\dot{\omega}_2)} \qquad (4—19)$$

8. 执行算法

由以上的数学模型，控制步进电机运动，在给定步数的情况下，速度从零才是加速，到达既定最大速度后开始匀速运动，运动到一定步数后开始减速，最后停下来到达给定的步数，速度曲线类似一个梯形的变化的过程，这样可以让电机启动或者停止更加平滑避免抖动的出现。见图 4-6。

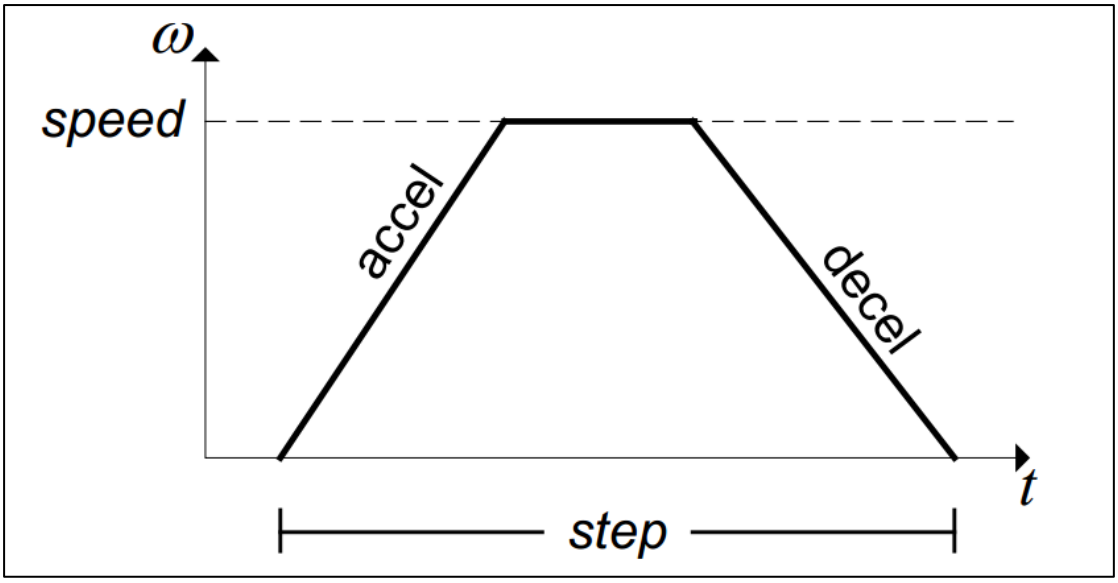

图 4-6 加减速曲线

速度属性的描述如下

step -移动的步数

accel -加速度

decal -减速度

speed -最大速度

为了让代码的运行速度提高，我们尽量避免浮点运算。因此为了提高计算精度，我们需要将一些数据进行放大后再进行运算。速度、加速度、减速度乘 100。将一些常量预先定义好，简化运算过程。

速度相关：

由式（4—1）、（4—2）、（4—3）、（4—4）可以得出

$$c_0 = \frac{n\alpha f_0}{\omega} \qquad (4—20)$$

这里ω为最大速度，用 speed 来表示，将它乘 100 倍，可以认为 speed 的单位为：0.01rad/s，f_0为定时器的频率，α为步距角，当 n=1 时可以求得c_0值即 min_delay 的值。

$$A\_T\_x100 = \alpha f_t \cdot 100 \qquad (4—21)$$

$$\min\_delay = \frac{A\_T\_x100}{speed} \qquad (4—22)$$

加速度相关：

由于数据推算过程的误差，这里c_0会乘 0.676 修正这个误差。

$$T1\_FREQ\_148 = 0.676 f_t/100 \qquad (4—23)$$

$$A\_SQ = 2\alpha \cdot 10000000000 \qquad (4—24)$$

由式（4—23）和（4—24）代入式（4—10）可以得出

$$step\_delay = c_0 = T1\_FREQ\_148\sqrt{\frac{A\_SQ}{accel}}/100 \qquad (4—25)$$

在加速的过程中，有两种场景计算速度属性：

1. 加速，直到达到所需的速度

2.未达到所需的速度就要开始减速

这场景取决于描述速度属性四个变量，首先介绍第一种场景，见图 4-7。

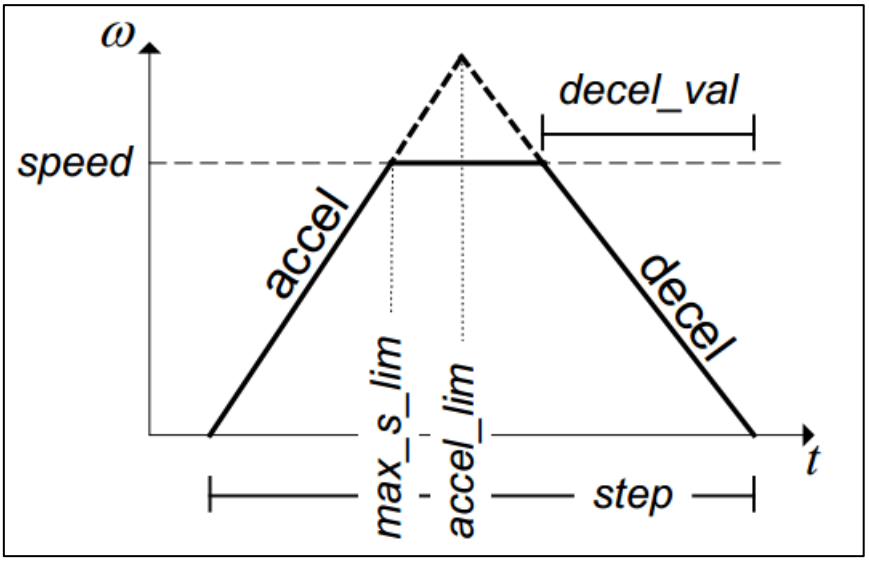

图 4-7 速度坡度受限于期望的速度

max_s_lim 是加速到所需的速度所需步数。

$$\text{max\_s\_lim} = \text{n} = \frac{speed^2}{2\alpha \cdot accel \cdot 100} \qquad (4—26)$$

accel_lim 是开始减速的步数。

$$\text{accel\_min} = n_1 = \frac{step \cdot decel}{accel + decel} \qquad (4—27)$$

如果 max_s_lim ＜ accel_lim,加速度是受限于最大的期望速度。由式（4—18）可以推出减速距离 decel_val 应该为，其中负号是因为加速和减速度的方向是相反的。

$$\text{decel\_val} = -\text{max\_s\_lim} \cdot \frac{accel}{decel} \qquad (4—28)$$

第二种场景是，运行的步数不足以进行加速到最大速度就要开始减速。见图 4-8。

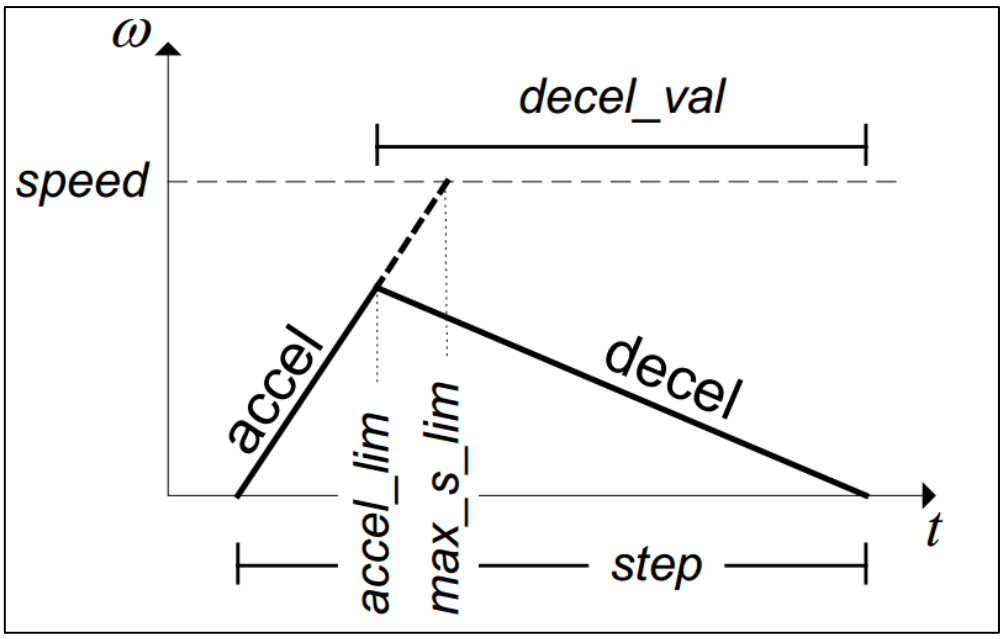

图 4-8 速度坡度未到达最大速度就开始减速

如果 max_s_lim ＞ accel_lim,加速度是受限于减速的开始。减速距离 decel_val 应该为

$$decel\_val = -(step - accel\_lim) \quad (4—29)$$

定时器中断处理函数

定时器中断产生步脉冲并且只有在步进电机移动时进入。这个中断处理速度属性的四个不同的状态，分别为 stop—accel—run—decel—stop。如图 4-9 所示。

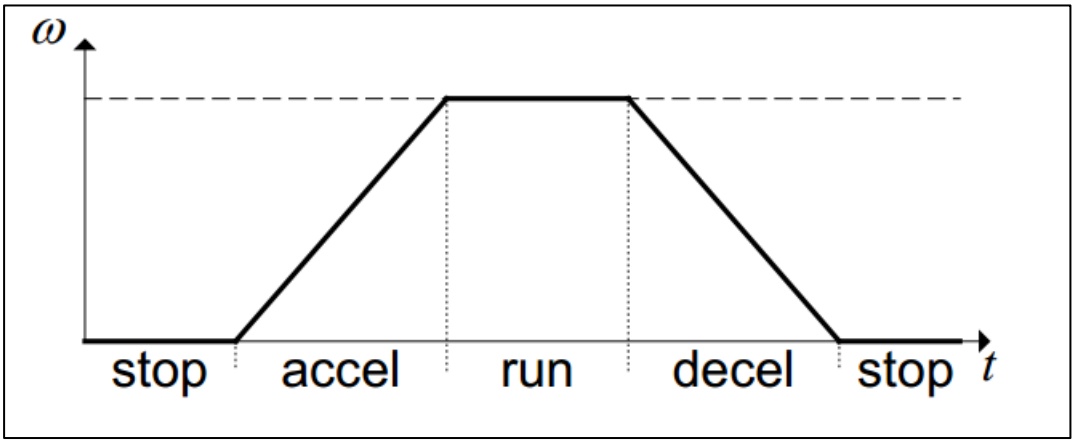

图 4-9 步进电机运行速度的四个状态

速度的这种行为通过状态机在定时器中断中实现。如图 4-10 所示。STOP 为停止状态，ACCEL 为加速状态，RUN 为匀速状态，DECEL 为减速状态。

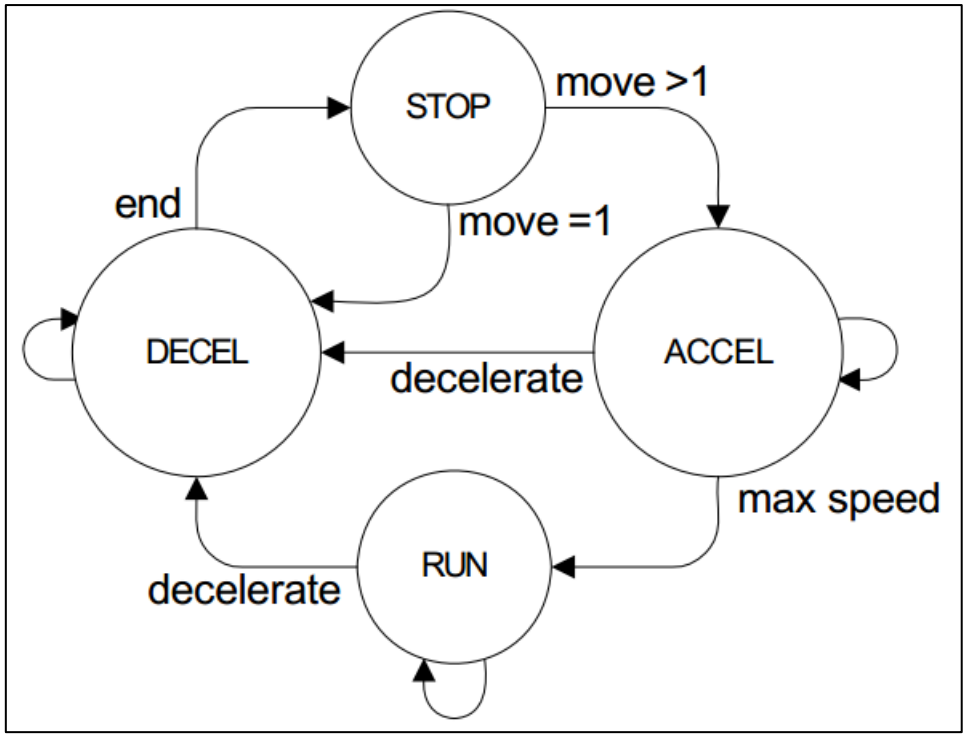

图 4-10 定时器中断中的状态机

当应用程序启动或者步进电机停止状态机处于 STOP 状态。当输入移动步数建立计算完成，一个新状态被置位同时定时器的中断被使能。当运行的步数超过 1 步状态机进入 ACCEL 状态，如果只移动一步，那么状态直接变为 DECEL。因为只移动一步是不需要加速的。当状态变为 ACCEL，应用程序一直加速步进电机达到期望的最大速度，这时状态变为 RUN，或者减速必须开始，状态变为 DECEL。它会一直保持 DECEL 状态并一直减速到期望的步数并且速度为 0。然后状态变为 STOP。

计算和定时器的计数器

在加速和减速过程中的每一步新的时间延迟必须计算得出。这个计算结果会包括一个系数和一个余数，为了提高精度余数保留并包含在下一次计算当中。由式（4—14）可以得出式（4—30），其中 rest 为余数，首次计算值为 0。new_rest 就是保存除不尽的余数参与下一次计算。

$$\text{new\_step\_delay} = \text{step\_delay} - \frac{2 \cdot step\_delay + rest}{4 \cdot accel\_count + 1} \quad （4—30）$$

$$\text{new\_rest} = \left(2 \cdot step_{delay} + rest\right)\left(mod(4 \cdot accel\_count + 1)\right) \quad （4—31）$$

当状态改变时为了保持位移的轨迹，一些辅助计数变量是必不可少的。见图 4-11。

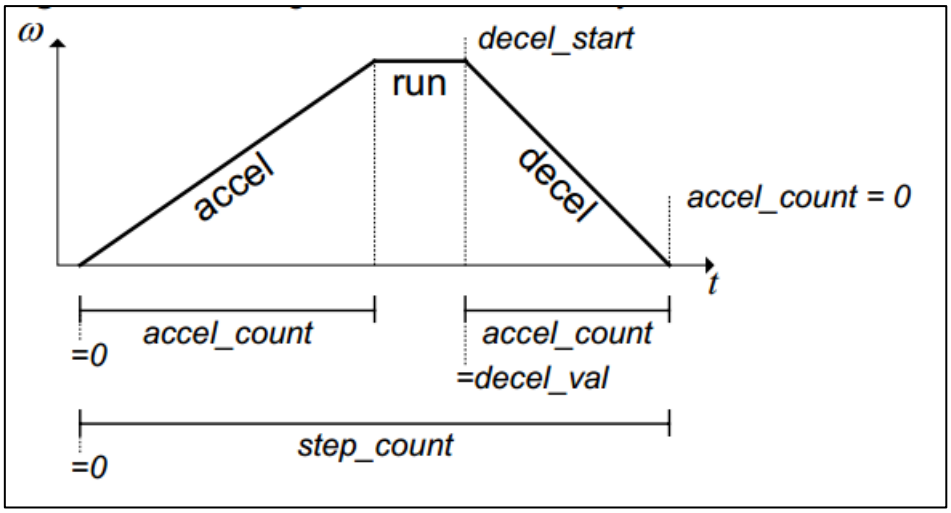

图 4-11 在延时中的计数变量

step_count 为计算步数，在 ACCEL 状态从零开始，在 DECEL 状态完成时结束。记录的步数应该与命令控制的步数相同。

accel_count 用于控制加速或者减速。在 ACCEL 状态时，它从零开始每一步都会增加直到 ACCEL 状态结束。在 DECEL 状态时，它设置为 decel_val，并且为负数，每一步都会增加直到它的值为 0，运动结束，状态设置为 STOP。

decel_start 指示减速开始。当 step_count 与 decel_start 相等时，状态设置为 DECEL。

当这个代码使用过程中我们需要特别注意中断的优先级。如果我们产生脉冲的中断被其他中断阻塞会影响电机的速度。

9. 代码讲解

配套的三个程序除了主控控制板不一样，其它部分是完全一样的，这里我们以“1. BH-MSD4805 测试程序 F103(指南者)”来讲解。

BH-MSD4805 的驱动包含了 MicroStepDriver.c 及 MSD_test.c 文件中。MicroStepDriver.c 文件中主要是配置定时器产生脉冲，以及实现步进电机移动的函数 MSD_Move(signed int step, unsigned int accel, unsigned int decel, unsigned int speed)。MSD_test.c 文件包含串口命令接收处理、控制步进电机运动。

MicroStepDriver.c 程序中首先初始化步进电机需要用到的三组信号：脉冲信号、方向信号、脱机信号。核心代码见代码清单 4-1。

代码清单 4-1 定时器初始化核心代码（位于 MicroStepDriver.c 和 MicroStepDriver.h 文件）

```
// 当使用不同的定时器的时候，对应的 GPIO 是不一样的，这点要注意
// 这里我们使用定时器 TIM2
#define              MSD_PULSE_TIM                    TIM2
#define          MSD_PULSE_TIM_APBxClock_FUN      RCC_APB1PeriphClockCmd
#define              MSD_PULSE_TIM_CLK                RCC_APB1Periph_TIM2
// 定时器输出 PWM 通道，PA0 是通道 1
#define              MSD_PULSE_OCx_Init               TIM_OC4Init
#define              MSD_PULSE_OCx_PreloadConfig      TIM_OC4PreloadConfig
// 定时器中断
#define              MSD_PULSE_TIM_IRQ                TIM2_IRQn
#define              MSD_PULSE_TIM_IRQHandler         TIM2_IRQHandler

// PWM 信号的频率 F = TIM_CLK/{(ARR+1)*(PSC+1)}
#define              MSD_PULSE_TIM_PERIOD             (10-1)
#define              MSD_PULSE_TIM_PSC                (72-1)

// 步进电机脉冲输出通道
#define              MSD_PULSE_GPIO_CLK               RCC_APB2Periph_GPIOA
#define              MSD_PULSE_PORT                   GPIOA
#define              MSD_PULSE_PIN                    GPIO_Pin_3

// 步进电机方向控制
#define              MSD_DIR_GPIO_CLK                 RCC_APB2Periph_GPIOB
#define              MSD_DIR_PORT                     GPIOB
#define              MSD_DIR_PIN                      GPIO_Pin_14

// 步进电机脉冲输出使能引脚
#define              MSD_ENA_GPIO_CLK                 RCC_APB2Periph_GPIOC
#define              MSD_ENA_PORT                     GPIOC
#define              MSD_ENA_PIN                      GPIO_Pin_4

static void TIM_Mode_Config(void)
{
    // 开启定时器时钟,即内部时钟 CK_INT=72M
    MSD_PULSE_TIM_APBxClock_FUN(MSD_PULSE_TIM_CLK,ENABLE);

    /*--------------------时基结构体初始化-------------------------*/
    TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
    TIM_TimeBaseStructInit(&TIM_TimeBaseStructure);
    // 自动重装载寄存器的值，累计 TIM_Period+1 个周期后产生一个更新或者中断
    TIM_TimeBaseStructure.TIM_Period=MSD_PULSE_TIM_PERIOD;
    // 驱动 CNT 计数器的时钟 = Fck_int/(psc+1)
    TIM_TimeBaseStructure.TIM_Prescaler= MSD_PULSE_TIM_PSC;
    // 时钟分频因子 ，配置死区时间时需要用到
    TIM_TimeBaseStructure.TIM_ClockDivision=TIM_CKD_DIV1;
    // 计数器计数模式，设置为向上计数
```

```
47      TIM_TimeBaseStructure.TIM_CounterMode=TIM_CounterMode_Up;
48      // 重复计数器的值，最大值为 255
49      //TIM_TimeBaseStructure.TIM_RepetitionCounter=0;
50      // 初始化定时器
51      TIM_TimeBaseInit(MSD_PULSE_TIM, &TIM_TimeBaseStructure);
52
53      /*--------------------输出比较结构体初始化-------------------*/
54      TIM_OCInitTypeDef  TIM_OCInitStructure;
55      // 配置为 PWM 模式 2
56      TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM2;
57      // 输出使能
58      TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
59      // 互补输出禁能
60      TIM_OCInitStructure.TIM_OutputNState = TIM_OutputNState_Disable;
61      // 设置占空比大小
62      TIM_OCInitStructure.TIM_Pulse = MSD_PULSE_TIM_PERIOD/2;
63      // 输出通道电平极性配置
64      TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_Low;
65      // 输出通道空闲电平极性配置
66      TIM_OCInitStructure.TIM_OCIdleState = TIM_OCIdleState_Reset;
67
68      MSD_PULSE_OCx_Init(MSD_PULSE_TIM, &TIM_OCInitStructure);
69      //使能 TIM1_CH1 预装载寄存器
70      MSD_PULSE_OCx_PreloadConfig(MSD_PULSE_TIM, TIM_OCPreload_Enable);
71      //使能 TIM1 预装载寄存器
72      TIM_ARRPreloadConfig(MSD_PULSE_TIM, ENABLE);
73
74      //设置中断源，只有溢出时才中断
75      TIM_UpdateRequestConfig(MSD_PULSE_TIM,TIM_UpdateSource_Regular);
76      // 清除中断标志位
77      TIM_ClearITPendingBit(MSD_PULSE_TIM, TIM_IT_Update);
78      // 使能中断
79      TIM_ITConfig(MSD_PULSE_TIM, TIM_IT_Update, ENABLE);
80      // 使能计数器
81      TIM_Cmd(MSD_PULSE_TIM, DISABLE);
82  }
```

MicroStepDriver.h 头文件主要定义驱动步进电机所需脉冲的定时器和它对应的输出通道引脚。方向信号引脚，脱机信号引脚。MicroStepDriver.c 程序中控制 STM32 使用 TIM2 定时器采用中断的方式产生脉冲，定时器的周期设定为 10us,占空比为 50%。使能溢出中断，使能计数器。

以给定的步数、加速度、减速度、最大速度移动步进电机。核心代码见代码清单 4-2。

代码清单 4-2 移动步进电机核心代码（位于 MicroStepDriver.c）

```
1  /*! \brief 以给定的步数移动步进电机
2   *  通过计算加速到最大速度，以给定的步数开始减速
3   *  如果加速度和减速度很小，步进电机会移动很慢，还没达到最大速度就要开始减速
4   *  \param step    移动的步数 (正数为顺时针，负数为逆时针).
5   *  \param accel   加速度,如果取值为 100，实际值为 100*0.01*rad/sec^2=1rad/sec^2
6   *  \param decel   减速度,如果取值为 100，实际值为 100*0.01*rad/sec^2=1rad/sec^2
7   *  \param speed   最大速度,如果取值为 100，实际值为 100*0.01*rad/sec=1rad/sec
8   */
9  void MSD_Move(signed int step, unsigned int accel, unsigned int decel, unsigned int speed)
10 {
11      //达到最大速度时的步数.
12      unsigned int max_s_lim;
13      //必须开始减速的步数(如果还没加速到达最大速度时)。
14      unsigned int accel_lim;
15
```

```
    // 根据步数的正负来判断方向
    if (step < 0) { //逆时针
        srd.dir = CCW;
        step = -step;
    } else { //顺时针
        srd.dir = CW;
    }
    // 输出电机方向
    DIR(srd.dir);
    // 配置电机为输出状态
    status.out_ena = TRUE;

    // 如果只移动一步
    if (step == 1) {
        // 只移动一步
        srd.accel_count = -1;
        // 减速状态
        srd.run_state = DECEL;
        // 短延时
        srd.step_delay = 1000;
        // 配置电机为运行状态
        status.running = TRUE;
        //设置定时器重装值
        TIM_SetAutoreload(MSD_PULSE_TIM,Pulse_width);
        //设置占空比为 50%
        TIM_SetCompare2(MSD_PULSE_TIM,Pulse_width>>1);
        //使能定时器
        TIM_Cmd(MSD_PULSE_TIM, ENABLE);
    }
    // 步数不为零才移动
    else if (step != 0) {
        // 我们的驱动器用户手册有详细的计算及推导过程

        // 设置最大速度极限，计算得到 min_delay 用于定时器的计数器的值。
        // min_delay = (alpha / tt)/ w
        srd.min_delay = A_T_x100 / speed;

        // 通过计算第一个(c0) 的步进延时来设定加速度，其中 accel 单位为 0.01rad/sec^2
        // step_delay = 1/tt * sqrt(2*alpha/accel)
        // step_delay = ( tfreq*0.676/100 )*100 * sqrt( (2*alpha*10000000000) / (accel*100) )/10000
        srd.step_delay = (T1_FREQ_148 * sqrt(A_SQ / accel))/100;

        // 计算多少步之后达到最大速度的限制
        // max_s_lim = speed^2 / (2*alpha*accel)
        max_s_lim = (long)speed*speed/(long)(((long)A_x20000*accel)/100);
        // 如果达到最大速度小于 0.5 步，我们将四舍五入为 0
        // 但实际我们必须移动至少一步才能达到想要的速度
        if (max_s_lim == 0) {
            max_s_lim = 1;
        }

        // 计算多少步之后我们必须开始减速
        // n1 = (n1+n2)decel / (accel + decel)
        accel_lim = ((long)step*decel) / (accel+decel);
        // 我们必须加速至少 1 步才能才能开始减速.
        if (accel_lim == 0) {
            accel_lim = 1;
        }
        // 使用限制条件我们可以计算出第一次开始减速的位置
        //srd.decel_val 为负数
        if (accel_lim <= max_s_lim) {
            srd.decel_val = accel_lim - step;
        } else {
            srd.decel_val = -(long)(max_s_lim*accel/decel);
```

```
80          }
81          // 当只剩下一步我们必须减速
82          if (srd.decel_val == 0) {
83              srd.decel_val = -1;
84          }
85
86          // 计算开始减速时的步数
87          srd.decel_start = step + srd.decel_val;
88
89          // 如果最大速度很慢，我们就不需要进行加速运动
90          if (srd.step_delay <= srd.min_delay) {
91              srd.step_delay = srd.min_delay;
92              srd.run_state = RUN;
93          } else {
94              srd.run_state = ACCEL;
95          }
96
97          // 复位加速度计数值
98          srd.accel_count = 0;
99          status.running = TRUE;
100          //设置定时器重装值
101          TIM_SetAutoreload(MSD_PULSE_TIM,Pulse_width);
102          //设置占空比为 50%
103          TIM_SetCompare2(MSD_PULSE_TIM,Pulse_width>>1);
104          //使能定时器
105          TIM_Cmd(MSD_PULSE_TIM, ENABLE);
106      }
107  }
```

MSD_Move(signed int step, unsigned int accel, unsigned int decel, unsigned int speed)函数作用是用户输入固定的步数、加速度、减速度、最大速度四个参数；然后计算出运动速度曲线，根据步数的正负来判断电机的方向；根据式（4—22）用最大速度来计算脉冲之间的最小延时对应定时器的计数值；根据式（4—25）计算出加速度的定时器计数值。根据式（4—26）用加速度和最大速度计算出到达最大速度是的步数；根据式（4—27）用步数、加速度和减速度计算必须开始减速的步数；根据给定的参数我们可以计算出电机是否能加速到最大速度就要开始减速，选择式（4—28）或者式（4—28）来计算减速距离。所有数据设置完毕后就启动定时器。

产生脉冲定时器的中断响应程序，每走一步都会重新计算运动状态。核心代码见代码清单 4-3。

代码清单 4-3 定时器中断服务函数核心代码（位于 MicroStepDriver.c）

```
1  /**
2
3    * @brief  产生脉冲定时器的中断响应程序，每走一步都会计算运动状态
4
5    * @param  无
6
7    * @retval 无
8
9    */
10 void MSD_PULSE_TIM_IRQHandler(void)
11 {
12     // 保存下一个延时周期
13     unsigned int new_step_delay;
14     // 加速过程中最后一次延时.
15     static int last_accel_delay;
16     // 移动步数计数器
17     static unsigned int step_count = 0;
18     // 记录 new_step_delay 中的余数，提高下一步计算的精度
```

```
    static signed int rest = 0;

    if (TIM_GetITStatus(MSD_PULSE_TIM, TIM_IT_Update) != RESET) {
        /* Clear MSD_PULSE_TIM Capture Compare1 interrupt pending bit*/
        TIM_ClearITPendingBit(MSD_PULSE_TIM, TIM_IT_Update);

        MSD_PULSE_TIM->CCR4=srd.step_delay >> 1;//周期的一半
        MSD_PULSE_TIM->ARR=srd.step_delay;
        //如果禁止输出，电机则停止运动
        if (status.out_ena != TRUE ||status.estop == TRUE) {
            srd.run_state = STOP;
        }
        switch (srd.run_state) {
        case STOP:
            step_count = 0;
            rest = 0;
            // Stop Timer/Counter 1.
            MSD_PULSE_TIM->CCER &= ~(1<<12); //禁止输出
            TIM_Cmd(MSD_PULSE_TIM, DISABLE);
            status.running = FALSE;
            break;

        case ACCEL:
            MSD_PULSE_TIM->CCER |= 1<<12; //使能输出
            MSD_StepCounter(srd.dir);
            step_count++;
            srd.accel_count++;
            new_step_delay = srd.step_delay - (((2 * (long)srd.step_delay)
                                                + rest)/(4 * srd.accel_count + 1));
            rest = ((2 * (long)srd.step_delay)+rest)%(4 * srd.accel_count + 1);
            // Chech if we should start decelration.
            if (step_count >= srd.decel_start) {
                srd.accel_count = srd.decel_val;
                srd.run_state = DECEL;
            }
            // 检查是否到达期望的最大速度
            else if (new_step_delay <= srd.min_delay) {
                last_accel_delay = new_step_delay;
                new_step_delay = srd.min_delay;
                rest = 0;
                srd.run_state = RUN;
            }
            break;

        case RUN:
            MSD_PULSE_TIM->CCER |= 1<<12; //使能输出
            MSD_StepCounter(srd.dir);
            step_count++;
            new_step_delay = srd.min_delay;
            // 检查是否需要开始减速
            if (step_count >= srd.decel_start) {
                srd.accel_count = srd.decel_val;
                // 以最后一次加速的延时作为开始减速的延时
                new_step_delay = last_accel_delay;
                srd.run_state = DECEL;
            }
            break;

        case DECEL:
            MSD_PULSE_TIM->CCER |= 1<<12; //使能输出
            MSD_StepCounter(srd.dir);
            step_count++;
            srd.accel_count++;
            new_step_delay = srd.step_delay - (((2 * (long)srd.step_delay)
                                               + rest)/(4 * srd.accel_count + 1));
            rest = ((2 * (long)srd.step_delay)+rest)%(4 * srd.accel_count + 1);
            // 检查是否为最后一步
```

```
86              if (srd.accel_count >= 0) {
87                  srd.run_state = STOP;
88              }
89              break;
90          }
91          srd.step_delay = new_step_delay;
92      }
93  }
94
```

中断服务函数中，使用一个状态机根据速度的四个状态来计算运动轨迹。根据式（4—30）和式（4—31）我们计算得出加速和减速时每一步需要脉冲的时间间隔。

串口中断服务函数中调用 DealSerialData 函数用作处理串口接收到的指令，核心代码见代码清单 4-4。

代码清单 4-4 USART1 串口接收数据处理核心代码（位于 MSD_test.c 文件）

```
1  /**
2
3    * @brief  处理串口接收到的数据
4
5    * @param  无
6
7    * @retval 无
8
9    */
10 void DealSerialData(void)
11 {
12     static char showflag =1;
13     // Number of steps to move.
14     static int steps = 40000;
15     // Accelration to use.
16     static int acceleration = 32000;
17     // Deceleration to use.
18     static int deceleration = 32000;
19     // Speed to use.
20     static int speed = 3000;
21
22     int acc_temp=0;
23     int dec_temp=0;
24     int speed_temp=0;
25
26     // Tells if the received string was a valid command.
27     char okCmd = FALSE;
28     if (showflag) {
29         showflag = 0;
30         ShowData(stepPosition, acceleration, deceleration, speed, steps);
31     }
32     // If a command is received, check the command and act on it.
33     if (status.cmd == TRUE) {
34         if (UART_RxBuffer[0] == 'm') {
35             // Move with...
36             if (UART_RxBuffer[1] == ' ') {
37                 // ...number of steps given.
38                 steps = atoi((char const *)UART_RxBuffer+2);
39                 MSD_Move(steps, acceleration, deceleration, speed);
40                 okCmd = TRUE;
41                 printf("\n\r  ");
42             } else if (UART_RxBuffer[1] == 'o') {
43                 if (UART_RxBuffer[2] == 'v') {
44                     if (UART_RxBuffer[3] == 'e') {
45                         // ...all parameters given
46                         if (UART_RxBuffer[4] == ' ') {
47                             int i = 6;
48                             steps = atoi((char const *)UART_RxBuffer+5);
49                  while ((UART_RxBuffer[i] != ' ') && (UART_RxBuffer[i] != 13)) i++;
50                             i++;
51                       acceleration = atoi((char const *)UART_RxBuffer+i);
```

```
52          while ((UART_RxBuffer[i] != ' ') && (UART_RxBuffer[i] != 13)) i++;
53                      i++;
54               deceleration = atoi((char const *)UART_RxBuffer+i);
55          while ((UART_RxBuffer[i] != ' ') && (UART_RxBuffer[i] != 13)) i++;
56                      i++;
57                      speed = atoi((char const *)UART_RxBuffer+i);
58                      MSD_Emergency_Stop(DISABLE);
59                       MSD_Move(steps, acceleration, deceleration, speed);
60                       okCmd = TRUE;
61                       printf("\n\r  ");
62                       }
63                   }
64                }
65            }
66          } else if (UART_RxBuffer[0] == 'a') {
67            // Set acceleration.
68            if (UART_RxBuffer[1] == ' ') {
69                acc_temp = atoi((char const *)UART_RxBuffer+2);
70                if (acc_temp>=71 && acc_temp<=32000) {
71                    acceleration = acc_temp;
72                    printf("\n\r 加速度:%.2frad/s^2",1.0*acceleration/100);
73                    okCmd = TRUE;
74                }
75            }
76          } else if (UART_RxBuffer[0] == 'd') {
77            // Set deceleration.
78            if (UART_RxBuffer[1] == ' ') {
79                dec_temp = atoi((char const *)UART_RxBuffer+2);
80                if (dec_temp>=71 && dec_temp<=32000) {
81                    deceleration = dec_temp;
82                    printf("\n\r 减速度:%.2frad/s^2",1.0*deceleration/100);
83                    okCmd = TRUE;
84                }
85            }
86          } else if (UART_RxBuffer[0] == 's') {
87            if (UART_RxBuffer[1] == ' ') {
88                speed_temp = atoi((char const *)UART_RxBuffer+2);
89                if (speed_temp>=12 && speed_temp<=20000) {
90                    speed = speed_temp;
91                    printf("\n\r 最大速度:%.2frad/s",1.0*speed/100);
92                    okCmd = TRUE;
93                }
94            }
95          } else if (UART_RxBuffer[0] == 13) {
96            MSD_Emergency_Stop(DISABLE);
97            MSD_Move(steps, acceleration, deceleration, speed);
98            okCmd = TRUE;
99          } else if (UART_RxBuffer[0] == '?') {
100              ShowHelp();
101              okCmd = TRUE;
102          } else if (UART_RxBuffer[0] == 't') {
103
104              {
105                  MSD_demo_run();
106              }
107              okCmd = TRUE;
108          }
109          // Send help if invalid command is received.
110          if (okCmd != TRUE) {
111              printf("\n\r 输入有误，请重新输入...");
112              ShowHelp();
113          }
114
115          // Clear RXbuffer.
116          status.cmd = FALSE;
117          uart_FlushRxBuffer();
118
```

```
119         if (status.running == TRUE) {
120             printf("\n\r 电机正在运行...");
121             while (status.running == TRUE) {
122                 if (status.estop)
123                     MSD_Emergency_Stop(ENABLE);
124             };
125             printf("OK\n\r");
126          ShowData(stepPosition, acceleration, deceleration, speed, steps);
127         }
128
129
130     }//end if(cmd)
131 }
```

串口接收处理函数中，主要处理串口中端发过来的数据进行处理，例如加速度 a,减速度 d,最大速度 s,运行步数进行设置。然后根据系统的状态状态判断电机的运行状态。串口终端打印每个动作的参数及最终的移动步数。

4.2 main 函数执行流程

main 函数代码较简单，它具体的执行流程如下：

- 初始化 USART1、板子上的 LED 灯、按键。
- 调用 MSD_Init 函数初始化步进电机相关 IO 口。
- 打印串口帮助命令 ShowHelp，方便用户根据提示输入指令。

接着在 while 循环里定时执行串口接收到数据的操作：

- 根据串口接收到的数据进行解包，判断是哪个指令并进行对应的操作。
- 若 KEY1 按按下，则转换驱动器切换为脱机模式，此时驱动器为无力矩保持，手动可以转动步进电机。
- 若 KEY2 被按下，则转换驱动器切换为正常工作模式，此时驱动器为有力矩保持，等待用户发送控制指令。

5. 产品更新及售后支持

秉火的产品资料更新会第一时间发布到论坛：http://www.firebbs.com

购买秉火产品请到秉火官方淘宝店铺：https://fire-stm32.taobao.com

在学习或使用秉火产品时遇到问题可在论坛发帖子与我们交流。