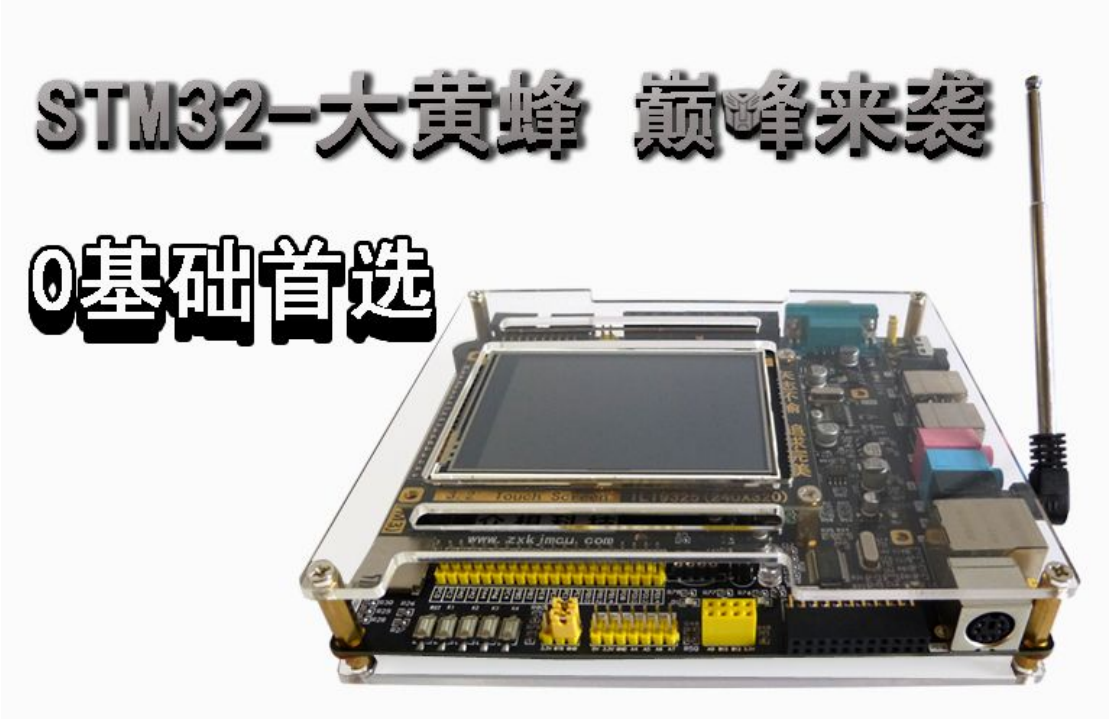
学 ARM 从 STM32 开始

STM32 开发板库函数教程--模块篇



官方网站：http://www.zxkjmcu.com
官方店铺：http://zxkjmcu.taobao.com
官方论坛：http://bbs.zxkjmcu.com
刘洋课堂：http://school.zxkjmcu.com

目 录

5.01 步进电机模块实验实验

5.01.1 步进电机概述

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（及步进角）。您可以通过控制脉冲个来控制角位移量，从而达到准确定位的目的；同时您可以通过控制脉冲频率来控制电机转动的速度和加速度，从而达到调速的目的。

步进电机 28BYJ48 型四相八拍电机，电压为 DC5V。当对步进电机施加一系列连续不断的控制脉冲时，它可以连续不断地转动。每一个脉冲信　号对应步进电机的某一相或两相绕组的通电状态改变一次，也就对应转子转过一定的角度（一个步距角）。当通电状态的改变完成一个循环时，转子转过一个齿距。四相步进电机可以在不同的通电方式下运行，常见的通电方式有单（单相绕组通电）四拍（A-B-C-D-A。。。），双（双相绕组通电）四拍（AB-BC-　CD-DA-AB-。。。），八拍（A-AB-B-BC-C-CD-D-DA-A。。。）

图 5.01.1　标准封装

5.03.1.1 步进电机性能描述

1、 步进电机必须加驱动才可以运转，驱动信号必须为脉冲信号，没有脉冲的时候，步进电机静止，如果加入适当的脉冲信号，就会以一定的角度（称为步角）转动。转动的速度和脉冲的频率成正比。

2、28BYJ485V 驱动的 4 相 5 线的步进电机，而且是减速步进电机，减速比为 1：64，步进角为 5.625/64 度。如果需要转动 1 圈，那么需要 360/5.625*64=4096 个脉冲信号。

3、步进电机具有瞬间启动和急速停止的优越特性。

4、改变脉冲的顺序，可以方便的改变转动的方向。

因此，目前打印机，绘图仪，机器人，等等设备都以步进电机为动力核心。

附表 2 ：主要技术参数

电机型号	28BYJ48	相电阻 ±10%	300Ω	启动转矩 100P.P.S.g.cm	≥300
电压 V	5	步距角度	5.625/64	启动频率 P.P.S	≥550
相数	4	减速比	1:64	定位转矩 g.cm	≥300
噪声 dB	≤35	绝缘介电强度	600VAC 1S		

5.01.1.2 拍数和步距角

四相步进电机有两种运行方式，一、四相四拍；二、四相八拍。 要想搞清楚四相八拍运行方式下步进电机的转速如果计算，需要先清楚两个基本概念。

1、拍数：完成一个磁场周期性变化所需脉冲数或导电状态用 n 表示，或指电机转过一个齿距角所需脉冲数，以四相电机为例，有四相四拍运行方式即“AB-BC-CD-DA-AB”，四相八拍运行方式“A-AB-B-BC-C-CD-D-DA-A”。

2、步距角：对应一个脉冲信号，电机转子转过的角位移用 θ 表示。

θ=360 度（转子齿数 J*运行拍数），以常规二、四相，转子齿为 50 齿电机为例。四拍运行时步距角为 θ=360 度/（50*4）=1.8 度（俗称整步），八拍运行时步距角为 θ=360 度/（50*8）=0.9 度（俗称半步）。

这两个概念清楚后，我们再来计算转速，以基本步距角 1.8° 的步进电机为例(现在市场上常规的二、四相混合式步进电机基本步距角都是 1.8°)，四相八拍运行方式下，每接收一个脉冲信号，转过 0.9° ，如果每秒钟接收 400 个脉冲，那么转速为每秒 400X0.9°=360° ，相当与每秒钟转一圈，每分钟 60 转。

附表 2 ：驱动方式（4-1-2 相驱动）

导线颜色	1	2	3	4	5	6	7	8
5 红色	+	+	+	+	+	+	+	+
4 橙色	-	-						-
3 黄色		-	-	-				
2 粉色				-	-	-		
1 蓝色						-	-	-

注释：红色接电源，橙色接 PD3，黄色接 PD6，粉色接 PD12，蓝色接 PE4。

5.01.2 实验目的及设备

一、实验目的

1. 掌握步进电机的工作原理；

2. 掌握单片机实现步进电机控制的基本方法，其中包括硬件和软件实现两部分；

3. 熟悉计算机测控系统中，步进电机作为控制对象的系统设计方法。

二、实验设备

大黄蜂系列 LY-STM32 型单片机开发板一套

PC 机一台

28BYJ-48 型步进电机一个

杜邦线若干

5.01.3 步进电机硬件设计

选用大黄蜂实验板，步进电机 28BYJ48 型四相八拍电机是成品模块，直接使用杜邦线插接到实验板上即可。硬件设计见“图 5.01.2　四相步进电机连线图”。

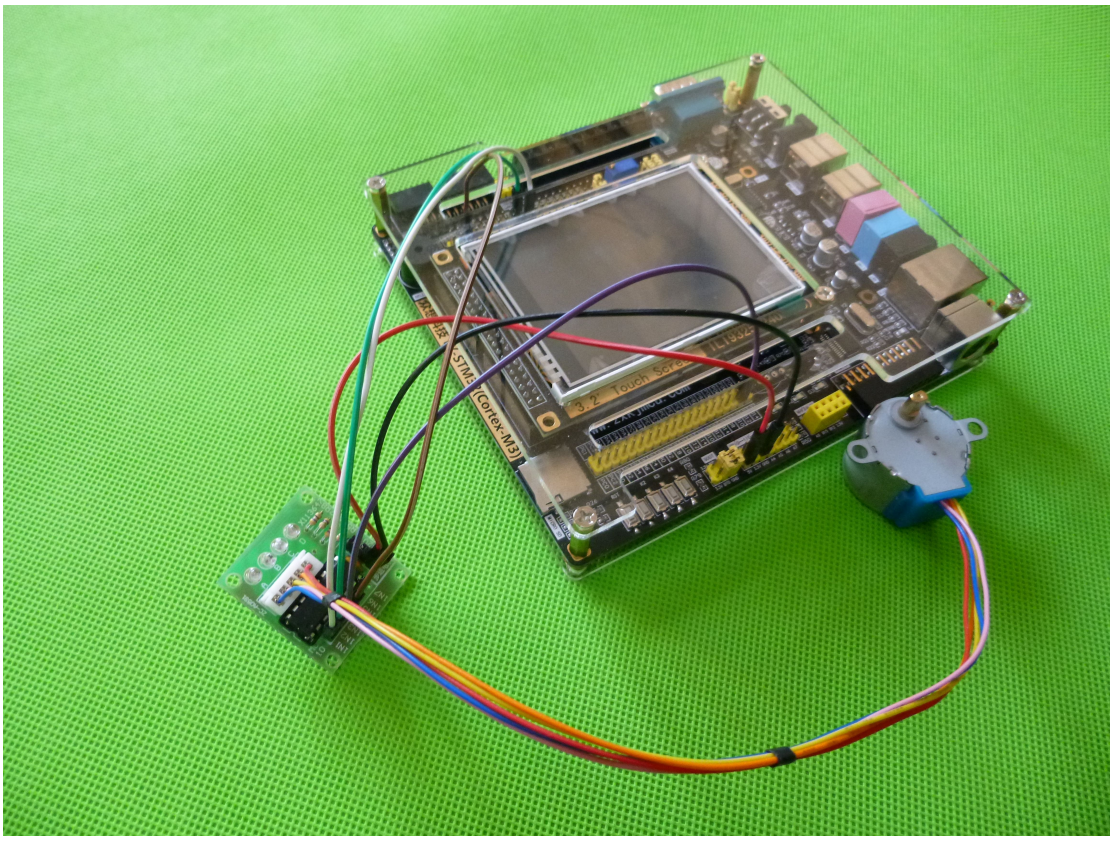


图 5.01.2　四相步进电机连线图

注释：从 K1 到 K4 分别是：K1--高速正转；K2--高速反转；K3--低速正转；K4--低速反转。

5.01.4 软件设计

5.01.4.1 软件设计说明

我们还是采用库函数的方式进行程序设计。在这节程序设计中，用到了外部中断函数；prinif 重定向打印输出函数； USART 串口通讯函数；定时器函数。

5.01.4.2 STM32 库函数文件

```
stm32f10x_gpio.c
stm32f10x_rcc.c
Misc.c  // 中断控制字（优先级设置）库函数
stm32f10x_exti.c   // 外部中断库处理函数
```

本节实验及以后的实验我们都是用到库文件，其中 stm32f10x_gpio.h 头文件包含了 GPIO 端口的定义。stm32f10x_rcc.h 头文件包含了系统时钟配置函数以及相关的外设时钟使能函数，所以我们要把这两个头文件对应的 stm32f10x_gpio.c 和 stm32f10x_rcc.c 加到工程中；Misc.c 库函数主要包含了中断优先级的设置，stm32f10x_exti.c 库函数主要包含了外部中断设置参数，这些函数也要添加到函数库中。以上库文件包含了本次实验所有要用到的函数功能。

5.01.5 main.c 文件里的内容是

```
#include "stm32f10x.h"
#include "stm32f10x_rcc.h"
#include "misc.h"

void RCC_Configuration(void);
```

```
    void GPIO_Configuration(void);
    void ZhengZhuan(u16 tt);
    void FanZhuan(u16 tt);
    void delay_ms(u16 nms);

    /*******************************************************************************
*****
    * 名    称：int main(void)
    * 功    能：主函数
    * 入口参数：无
    * 出口参数：无
    * 说    明：
    * 调用方法：无
    *******************************************************************************
****/
    int main(void)
    {
      RCC_Configuration();  //系统时钟设置及外设时钟使能
      GPIO_Configuration();

      while (1)
      {
       //读取 PC5 管脚的输入状态     K1
       if (GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_5) == Bit_RESET)
       {
          ZhengZhuan(5);//高速
       }

       //读取 PC5 管脚的输入状态     K2
       if (GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_1) == Bit_RESET)
       {
          FanZhuan(5);//高速
       }

       //读取 PC2 管脚的输入状态         K3
       if (GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_2) == Bit_RESET)
       {
          ZhengZhuan(10);//低速
       }

       //读取 PC3 管脚的输入状态         K4
       if (GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_3) == Bit_RESET)
       {
          FanZhuan(10);//低速
```

```
      }

        GPIO_ResetBits(GPIOD, GPIO_Pin_3);
        GPIO_ResetBits(GPIOD, GPIO_Pin_6);
        GPIO_ResetBits(GPIOD, GPIO_Pin_12);
        GPIO_ResetBits(GPIOE, GPIO_Pin_4);
      }
    }

/*******************************************************************************
*
* 名    称：void RCC_Configuration(void)
* 功    能：系统时钟配置为 72MHZ， 外设时钟配置
* 入口参数：无
* 出口参数：无
* 说    明：
* 调用方法：无
*******************************************************************************/
void RCC_Configuration(void)
{
  SystemInit();

RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC|RCC_APB2Periph_GPIOD|RCC_APB2Per
iph_GPIOE , ENABLE);
}

/*******************************************************************************
* 名    称：void GPIO_Configuration(void)
* 功    能：LED 控制口线及键盘设置
* 入口参数：无
* 出口参数：无
* 说    明：
* 调用方法：无
*******************************************************************************
/
void GPIO_Configuration(void)
{
  GPIO_InitTypeDef GPIO_InitStructure;            //端口配置结构体

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;          //PD3 管脚
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;   //推挽输出
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;  //口线翻转速度为 50MHz
  GPIO_Init(GPIOD, &GPIO_InitStructure);          //初始化端口
```

```
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6;          //PD6 管脚
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;   //推挽输出
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;  //口线翻转速度为 50MHz
  GPIO_Init(GPIOD, &GPIO_InitStructure);          //初始化端口

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;      //PD12 管脚
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;   //推挽输出
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;  //口线翻转速度为 50MHz
  GPIO_Init(GPIOD, &GPIO_InitStructure);          //初始化端口

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4;          //PE4 管脚
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;   //推挽输出
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;  //口线翻转速度为 50MHz
  GPIO_Init(GPIOE, &GPIO_InitStructure);          //初始化端口


  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;          //PC5 管脚
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;      //输入上拉
  GPIO_Init(GPIOC, &GPIO_InitStructure);          //初始化端口

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;          //PC1 管脚
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;      //输入上拉
  GPIO_Init(GPIOC, &GPIO_InitStructure);          //初始化端口

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;          //PC2 管脚
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;      //输入上拉
  GPIO_Init(GPIOC, &GPIO_InitStructure);          //初始化端口

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;          //PC3 管脚
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;      //输入上拉
  GPIO_Init(GPIOC, &GPIO_InitStructure);          //初始化端口
}
//电机正转函数
void ZhengZhuan(u16 tt)
{
     //1100
     GPIO_SetBits(GPIOD, GPIO_Pin_3);
     GPIO_SetBits(GPIOD, GPIO_Pin_6);
     GPIO_ResetBits(GPIOD, GPIO_Pin_12);
     GPIO_ResetBits(GPIOE, GPIO_Pin_4);
     delay_ms(tt);
     //0110
     GPIO_ResetBits(GPIOD, GPIO_Pin_3);
     GPIO_SetBits(GPIOD, GPIO_Pin_6);
```

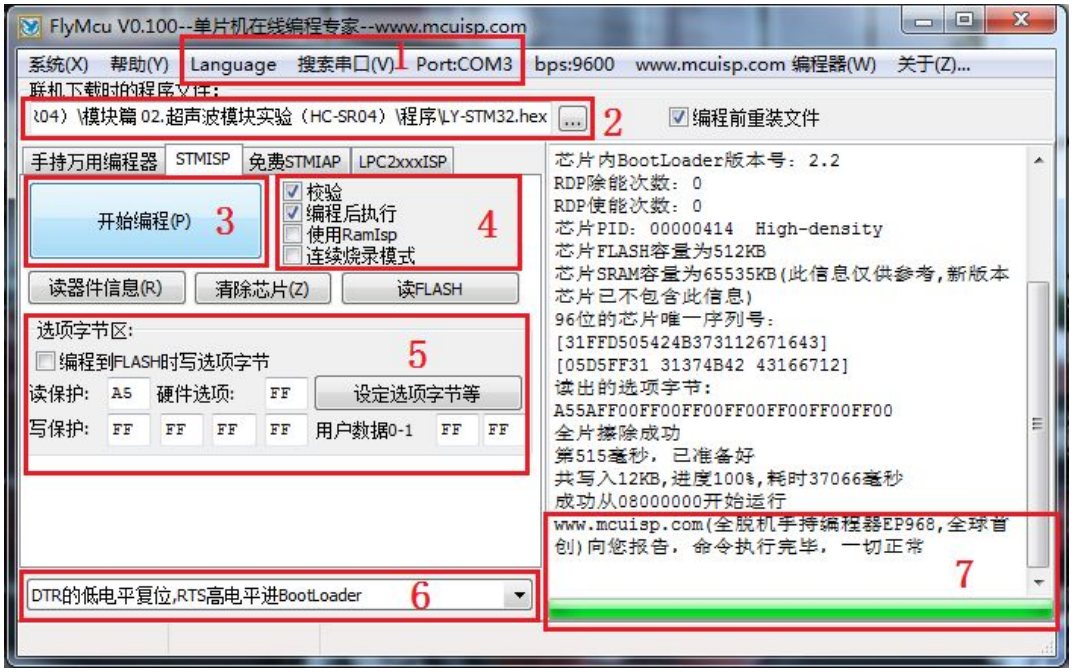
```
	GPIO_SetBits(GPIOD, GPIO_Pin_12);
	GPIO_ResetBits(GPIOE, GPIO_Pin_4);
	delay_ms(tt);
	//0011
	GPIO_ResetBits(GPIOD, GPIO_Pin_3);
	GPIO_ResetBits(GPIOD, GPIO_Pin_6);
	GPIO_SetBits(GPIOD, GPIO_Pin_12);
	GPIO_SetBits(GPIOE, GPIO_Pin_4);
	delay_ms(tt);
	//1001
	GPIO_SetBits(GPIOD, GPIO_Pin_3);
	GPIO_ResetBits(GPIOD, GPIO_Pin_6);
	GPIO_ResetBits(GPIOD, GPIO_Pin_12);
	GPIO_SetBits(GPIOE, GPIO_Pin_4);
	delay_ms(tt);
}
//电机反转函数
void FanZhuan(u16 tt)
{
	//1001
	GPIO_SetBits(GPIOD, GPIO_Pin_3);
	GPIO_ResetBits(GPIOD, GPIO_Pin_6);
	GPIO_ResetBits(GPIOD, GPIO_Pin_12);
	GPIO_SetBits(GPIOE, GPIO_Pin_4);
	delay_ms(tt);
	//0011
	GPIO_ResetBits(GPIOD, GPIO_Pin_3);
	GPIO_ResetBits(GPIOD, GPIO_Pin_6);
	GPIO_SetBits(GPIOD, GPIO_Pin_12);
	GPIO_SetBits(GPIOE, GPIO_Pin_4);
	delay_ms(tt);
	//0110
	GPIO_ResetBits(GPIOD, GPIO_Pin_3);
	GPIO_SetBits(GPIOD, GPIO_Pin_6);
	GPIO_SetBits(GPIOD, GPIO_Pin_12);
	GPIO_ResetBits(GPIOE, GPIO_Pin_4);
	delay_ms(tt);
	//1100
	GPIO_SetBits(GPIOD, GPIO_Pin_3);
	GPIO_SetBits(GPIOD, GPIO_Pin_6);
	GPIO_ResetBits(GPIOD, GPIO_Pin_12);
	GPIO_ResetBits(GPIOE, GPIO_Pin_4);
	delay_ms(tt);
}
```

```
void delay_ms(u16 nms)
{
    u32 temp;
    SysTick->LOAD = 9000*nms;
    SysTick->VAL=0X00;//清空计数器
    SysTick->CTRL=0X01;//使能，减到零是无动作，采用外部时钟源
    do
    {
     temp=SysTick->CTRL;//读取当前倒计数值
    }while((temp&0x01)&&(!(temp&(1<<16))));//等待时间到达
    SysTick->CTRL=0x00; //关闭计数器
    SysTick->VAL =0X00; //清空计数器
}
```

5.01.6 程序下载

在这一章节中要掌握 DHT11 温湿度传感器工作时序，了解常用的温湿度传感器功能和原理。

请根据下图所指向的 7 个重点区域配置。其中（1）号区域根据自己机器的实际情况选择，我的机器虚拟出来的串口号是 COM3。（2）号区域请自己选择程序代码所在的文件夹。（7）号区域当程序下载完后，进度条会到达最右边，并且提示一切正常。（4、5、6）号区域一定要按照上图显示的设置。当都设置好以后就可以直接点击（3）号区域的开始编程按钮下传程序了。



本节实验的源代码在光盘中：（LY-STM32 光盘资料\1.课程\2,外设篇\模块篇 1.步进电机模块实验（28BYJ48）\程序）

5.01.7 步进电机小常识

1.什么是步进电机?

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（及步进角）。您可以通过控制脉冲个来控制角位移量，从而达到准确定位的目的；同时您可以通过控制脉冲频率来控制电机转动的速度和加速度，从而达到调速的目的。

2.步进电机分哪几种?

步进电机分三种：永磁式（PM），反应式（VR）和混合式（HB）

永磁式步进一般为两相，转矩和体积较小，步进角一般为 7.5 度或 15 度；

反应式步进一般为三相，可实现大转矩输出，步进角一般为 1.5 度，但噪声

和振动都很大。在欧美等发达国家 80 年代已被淘汰；混合式步进是指混合了永磁式和反应式的优点。它又分为两相和五相：两相步进角一般为 1.8 度而五相步进角一般为 0.72 度。这种步进电机的应用最为广泛。

3.什么是保持转矩（HOLDING TORQUE）？

保持转矩（HOLDING TORQUE）是指步进电机通电但没有转动时，定子锁住转子的力矩。它是步进电机最重要的参数之一，通常步进电机在低速时的力矩接近保持转矩。由于步进电机的输出力矩随速度的增大而不断衰减，输出功率也随速度的增大而变化，所以保持转矩就成为了衡量步进电机最重要的参数之一。比如，当人们说 2N.m 的步进电机，在没有特殊说明的情况下是指保持转矩为 2N.m 的步进电机。

6.步进电机的外表温度允许达到多少?

步进电机温度过高首先会使电机的磁性材料退磁，从而导致力矩下降乃至于失步，因此电机外表允许的最高温度应取决于不同电机磁性材料的退磁点；一般来讲，磁性材料的退磁点都在摄氏 130 度以上，有的甚至高达摄氏 200 度以上，所以步进电机外表温度在摄氏 80-90 度完全正常。

7.为什么步进电机的力矩会随转速的升高而下降?

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；频率越高，反向电动势越大。在它的作用下，电机随频率（或速度）的增大而相电流减小，从而导致力矩下降。

8.为什么步进电机低速时可以正常运转,但若高于一定速度就无法启动,并伴有啸叫声?

步进电机有一个技术参数：空载启动频率，即步进电机在空载情况下能

够正常启动的脉冲频率，如果脉冲频率高于该值，电机不能正常启动，可能发生丢步或堵转。在有负载的情况下，启动频率应更低。如果要使电机达到高速转动，脉冲频率应该有加速过程，即启动频率较低，然后按一定加速度升到所希望的高频（电机转速从低速升到高速）。

9.如何克服两相混合式步进电机在低速运转时的振动和噪声?

步进电机低速转动时振动和噪声大是其固有的缺点，一般可采用以下方案来克服：

A.如步进电机正好工作在共振区，可通过改变减速比等机械传动避开共振区；

B.采用带有细分功能的驱动器，这是最常用的、最简便的方法；

C.换成步距角更小的步进电机，如三相或五相步进电机；

D.换成交流伺服电机，几乎可以完全克服震动和噪声，但成本较高；

E.在电机轴上加磁性阻尼器，市场上已有这种产品，但机械结构改变较大。

10.细分驱动器的细分数是否能代表精度?

步进电机的细分技术实质上是一种电子阻尼技术（请参考有关文献），其主要目的是减弱或消除步进电机的低频振动，提高电机的运转精度只是细分技术的一个附带功能。比如对于步进角为 1.8° 的两相混合式步进电机，如果细分驱动器的细分数设置为 4，那么电机的运转分辨率为每个脉冲 0.45°，电机的精度能否达到或接近 0.45°，还取决于细分驱动器的细分电流控制精度等其它因素。不同厂家的细分驱动器精度可能差别很大；细分数越大精度越难控制。

11.四相混合式步进电机与驱动器的串联接法和并联接法有什么区别?

四相混合式步进电机一般由两相驱动器来驱动，因此，连接时可以采用串联接法　或并联接法将四相电机接成两相使用。串联接法一般在电机转速较的场合使用，　此时需要的驱动器输出电流为电机相电流的 0.7 倍，因而电机发热小；并联接法　一般在电机转速较高的场合使用（又称高速接法），所需要的驱动器输出电流为　　电机相电流的 1.4 倍,因而电机发热较大。

12.如何确定步进电机驱动器的直流供电电源?

A.电压的确定

混合式步进电机驱动器的供电电源电压一般是一个较宽的范围（比如 IM483 的供电电压为 12～48VDC），电源电压通常根据电机的工作转速和响应要求来选择。如果电机工作转速较高或响应要求较快,那么电压取值也高,但注意电源电压的纹波不能超过驱动器的最大输入电压,否则可能损坏驱动器。

B.电流的确定

供电电源电流一般根据驱动器的输出相电流 I 来确定。如果采用线性电源，电源电流一般可取 I 的 1.1～1.3 倍；如果采用开关电源，电源电流一般可取 I 的 1.5～2.0 倍。

13.混合式步进电机驱动器的脱机信号 FREE 一般在什么情况下使用?

当脱机信号 FREE 为低电平时，驱动器输出到电机的电流被切断，电机转子处于自由状态（脱机状态）。在有些自动化设备中，如果在驱动器不断电的情况下要求直接转动电机轴（手动方式），就可以将 FREE 信号置低，使电机脱机，进行手动操作或调节。手动完成后，再将 FREE 信号置高，以

继续自动控制。

14. 如果用简单的方法调整两相步进电机通电后的转动方向?

只需将电机与驱动器接线的 A+和 A-（或者 B+和 B-）对调即可。