WHEELTEC
INTELLIGENT TECHNOLOGY

轮 趣 科 技

步进电机双路驱动底板

D302A 使用手册

推荐关注我们的公众号获取更新资料

版本说明：

版本	日期	内容说明
V1.0	2023/05/08	第一次发布

网址：www.wheeltec.net

序言

我们通过这篇教程与大家一起学习步进电机双路驱动底板（D302A）的使用。本驱动底板使用的芯片为 LV8731V 步进电机驱动芯片。

目录

1. 步进电机入门基础知识

1.1 步进电机原理

步进电机是将脉冲信号转化为角位移或线位移的开环控制电机，在非超载的情况下，电机的转速、停止位置只取决与脉冲的频率，而不受负载变化的影响。当步进驱动器收到一个脉冲信号，它就可以驱动电机按设定的方向转动一个固定的角度，称为步距角。步进电机的旋转是以固定的角度一步一步运行的，可以通过控制脉冲的个数来控制角位移量，从而达到准确的定位，同时也可以通过控制脉冲的频率来控制电机的速度和加速度。但在实际工作中，电机旋转的步距角会有微小的差距，主要是由电机结构上的固定有误差产生的，而且这种误差不会累计。

步进电机主要按照转子的特点和定子绕组进行分裂，按照转子的分类主要分为反应式、永磁式和混合式；按照定子分类主要分为两相、三相、五相等。目前最受欢迎的就是两相混合式步进电机，42 步进电机就是这种，这种电机的基本步距角为 1.8 度一步，配合上细分驱动器之后，可以让步进电机的旋转步距角更小且运行的更加平稳。

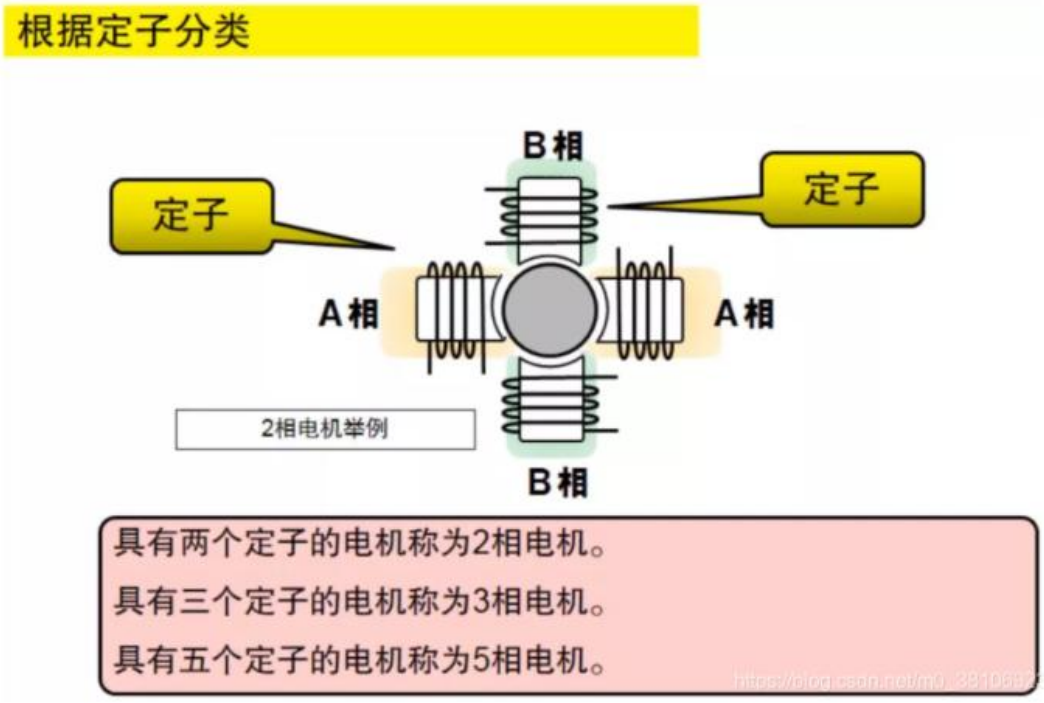

图 1 电机定子示意图

我们做这样的一个实验就可以让步进电机转动起来。1 找一节电池正负随意接入到 A 相两端;然后断开;(记为 A 正向)2 再将电池接入到 B 相两端; 然后断开;(记为 B 正向)3 电池正负对调再次接入 A 相; 然后断开;(记为 A 负向)4 保持正负对调接入 B 相;然后断开;(记为 B 负向)…如此循环你会看到步进电机在缓慢转动。**注意电机的相电阻是很小的接通时近乎短路**。我们将相电流的方向记录下来应该为:A+B+A-B-A+…，如果我们更换接线顺序使得相电流顺序为 A+B-A-B+A+…这时我们会看到电机向反方向运动。

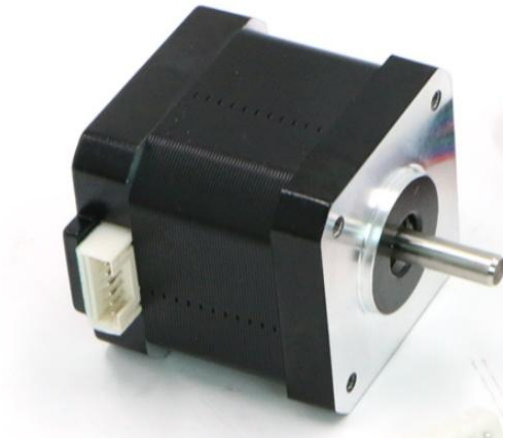

图 2 42 步进电机示意图

2.驱动底板的介绍

2.1 功能特性

- 单路典型最大电流 2.5A，典型值在 1.5A 左右，特别适合驱动 42 步进电机。
- 内置 XL2596_5.0 电源芯片，可为外部提供 5V 输出，
- 板子上预留 4 个 3mm 直径的安装圆孔，便于固定。
- 板子布线布局好，过电流能力强，同时也更有利于散热。
- 接口文字说明清晰，使用方便。
- 基于默认的 16 细分，控制大部分的 42 步进电机，推荐输入的控制频率范围：0~13KHz。
- 驱动板子工作温度范围：-10°~55°
- 工作温度范围：-20°~85°，实际使用中控制芯片最大温度不要超过 105°。
- 安装尺寸：60*99mm
- 驱动电压：9V~32V

2.2 应用场合

- 自动化机器人
- 机电一体化
- 设备制造
- 科研、生产
- 电子竞赛

2.3 综合描述

本步进电机驱动底板是专为 42 步进电机设计，电机芯片使用的是 LV8731V，每通道有峰值为 2.5A 的电流输出，但是推荐使用的为 1.5A 电流。驱动底板拥有良好的设计思路，可以直接插入本公司设计的最小核心板 CO3C 控制步进电机。并有多个引出的模块接口，便于二次开发和使用。默认不焊接 CAN 驱动芯片 U2，若想利用，可以自行购买 VP230 焊接上。

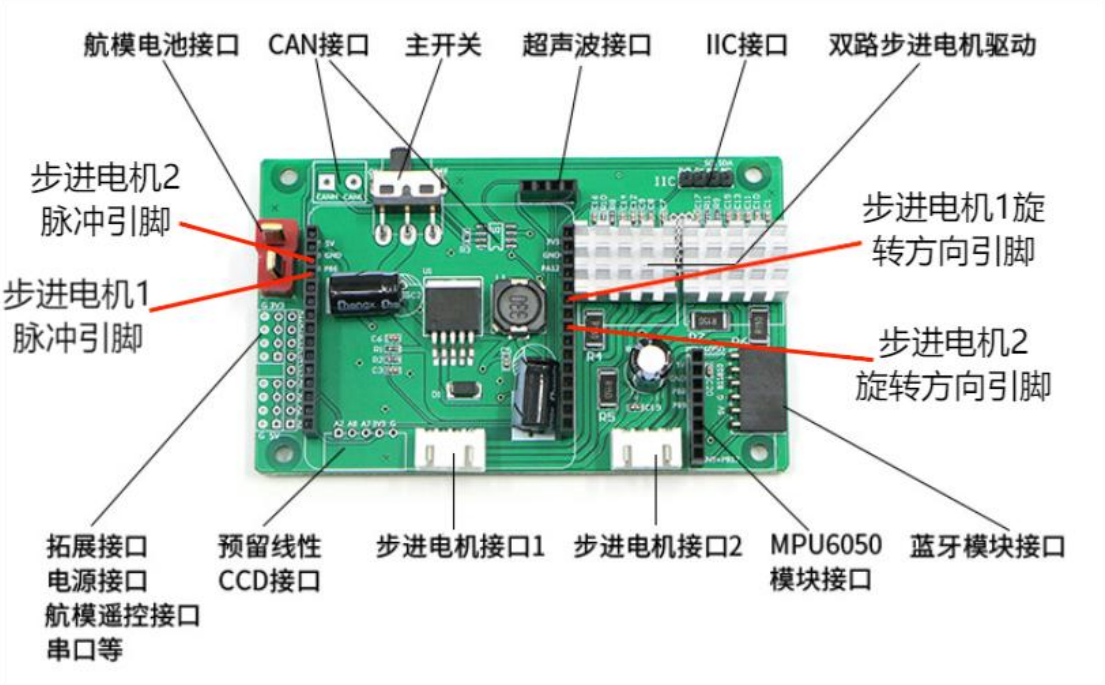

图 3 驱动底板详细接口示意图

3. 原理图说明

3.1 LV8731V 芯片原理图介绍与控制说明

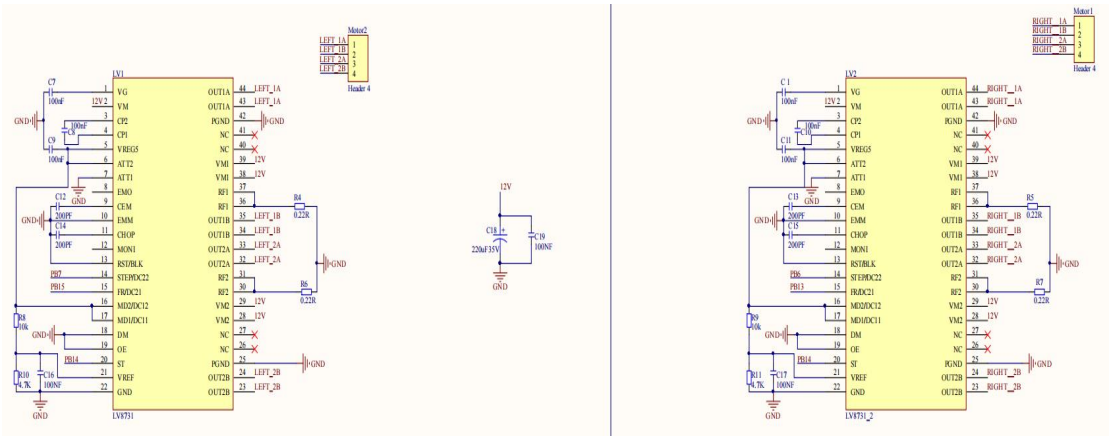

图 4 LV8731 芯片原理图

表 1 LV8731V 主要引脚表

VM	电机电源连接端子
VG	电荷泵电容连接端子
CP1、CP2	电荷泵电容连接端子
VREF	定电流控制基准电压输入端子
STEP	脉冲信号输入端子
FR	CW/CCW 信号输入端子
OUT1A/2A	A 相输出端子
OUT1B/2B	B 相输出端子
ST	CHIP 有效端子，控制电机运动或待机状态
DM	驱动模式切换端子，低电平为步进电机 1CH
EMM	切换短路保护模式端子
PGND	电源地
CEM	输出短路状态检出时间设定电容连接端子
VREG5	内部电源用电容连接端子
CHOP	间歇频率用电容连接端子
RTS	RESET 连接端子

在使用本驱动板的过程中，只需要通过板子的 T 头供电即可，然后用驱动板上的 5V 或者 3V3 给单片机供电，如购买了本公司的 C03C 最小核心板，按照正确的引脚指示插上即可，接着通过单片机给驱动板发送脉冲信号和电机旋转方向的信号即可正常驱动电机。FR 引脚是控制电机正反转的引脚，低电平顺时针转动，高电平逆时针转动。

4. 接线说明

提供的接线说明都是基于本公司产品的例程，我们提供了 STM32F103C8T6 核心板的例程一份。别的产品需要自己摸索一下，但也只需改动引脚，只要是和本例程功能相同的引脚即可。

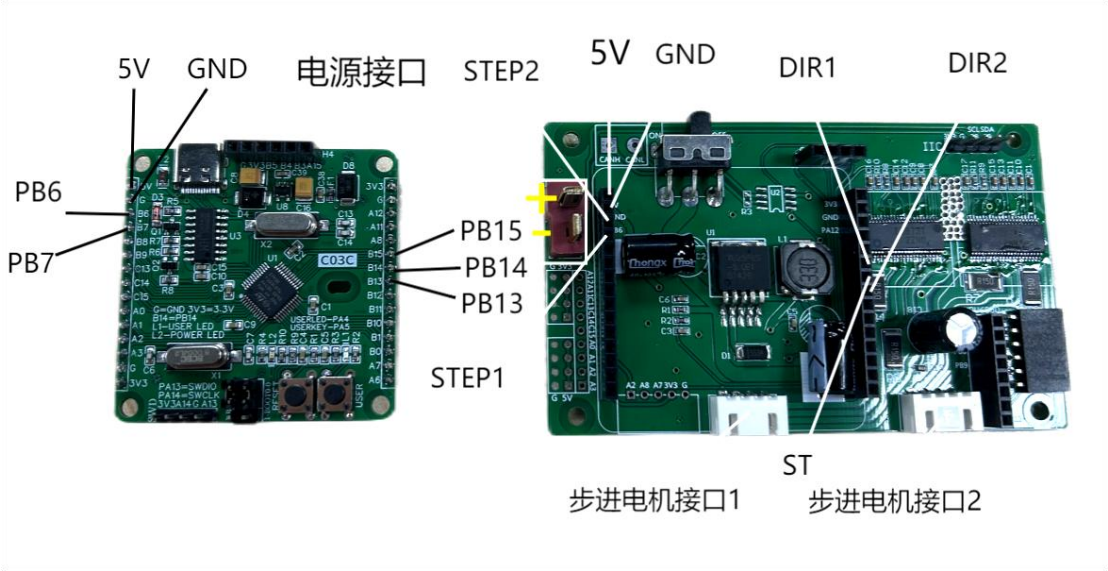

图 5 接线示意图

本例程使用的是 C03C 核心板，如已购买，可直接插上使用。

表 2 STM32 与驱动板接线表

C8T6	驱动板	外部
	电源接口+	12V 电源+
	电源接口-	12V 电源-
5V	5V	
GND	GND	
PB6	STEP2	
PB7	STEP1	
PB15	DIR1	
PB13	DIR2	
PB14	ST	
	电机接口 2	步进电机
	电机接口 1	步进电机

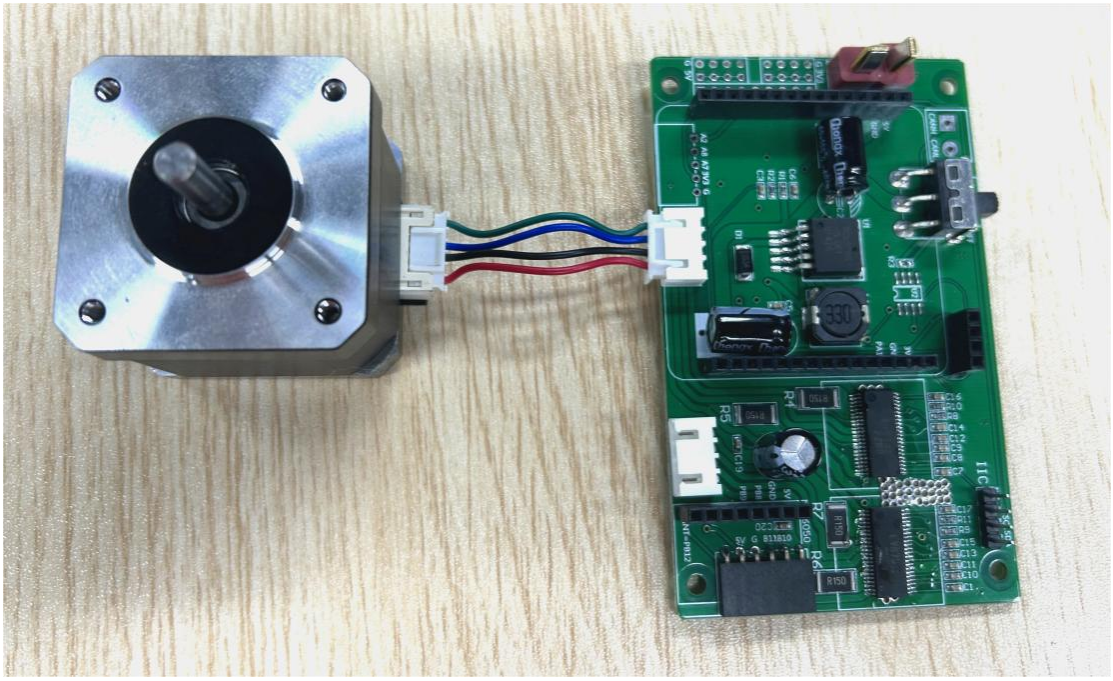

图 6 步进电机与驱动板接线示意图

5. 例程代码

5.1 电机 DIR 和 ST 引脚的初始化

```
/**************************************************************************
函数功能：初始化DIR和ST引脚
入口参数：无
返回  值：无
**************************************************************************/
void Gpio_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;                //定义结构体GPIO_InitStructure

    RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOB, ENABLE); // 使能PB端口时钟
    GPIO_InitStructure.GPIO_Pin =    GPIO_Pin_15 |GPIO_Pin_14|GPIO_Pin_13;      //PB15、PB13
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;          //推挽，增大电流输出能力
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;   //IO口速度
    GPIO_Init(GPIOB, &GPIO_InitStructure);            //GBIOA初始化
    GPIO_SetBits(GPIOB,GPIO_Pin_14);                                    //置高使能PB14

}
```

图 7 电机 DIR 和 ST 引脚初始化

这是对电机的控制方向引脚和使能引脚初始化，并且拉高 PB14 引脚，也就是 ST 引脚，使能电机驱动芯片，使芯片正常工作。

5.2 脉冲引脚初始化

```
void pwm_int_TIM4(u16 arr,u16 psc)
{
    GPIO_InitTypeDef GPIO_InitStructure;                //定义结构体GPIO_InitStructure
    TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;       //定义结构TIM_TimeBaseStructure
    NVIC_InitTypeDef NVIC_InitStructure;                        //定义结构NVIC_InitStructure
    TIM_OCInitTypeDef  TIM_OCInitStructure;               //定义结构TIM_OCInitStructure

    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4,ENABLE);  //使能定时器4时钟
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);//使能PB端口时钟

    GPIO_InitStructure.GPIO_Pin =    GPIO_Pin_6 | GPIO_Pin_7;          //PB6 PB7
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;     //IO口速度
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;         //复用模式输出
    GPIO_Init(GPIOB, &GPIO_InitStructure);            //GBIOB初始化

    TIM_TimeBaseStructure.TIM_Period = arr;    //设置在下一个更新事件装入活动的自动重装载寄存器周期的值
    TIM_TimeBaseStructure.TIM_Prescaler = psc; //设置用来作为TIMx时钟频率除数的预分频值   不分频
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;  //设置时钟分割:TDTS = Tck_tim
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up; //TIM向上计数模式

    TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);//根据TIM_TimeBaseInitStruct中指定的参数初始化TIMx的时间基数单位

    TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_Toggle;    //PWMTIM脉冲翻转模式
    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable; //比较输出使能
    TIM_OCInitStructure.TIM_Pulse = 0; //设置待装入捕获比较寄存器的脉冲值
    TIM_OCInitStructure.TIM_OCPolarity = TIM_CounterMode_Up;        //设置TIM输出比较极性为高
    TIM_OCInitStructure.TIM_OutputState=    TIM_OutputState_Enable;   //比较输出使能

    TIM_OC1Init(TIM4, &TIM_OCInitStructure);  //根据TIM_OCInitStructure中指定的参数初始化外设TIM4
    TIM_OC1PreloadConfig(TIM4, TIM_OCPreload_Disable);      //使能预装载寄存器

    TIM_OC2Init(TIM4, &TIM_OCInitStructure);  //根据TIM_OCInitStructure中指定的参数初始化外设TIM4
    TIM_OC2PreloadConfig(TIM4, TIM_OCPreload_Disable);      //使能预装载寄存器

    TIM_ARRPreloadConfig(TIM4, ENABLE);      //使能自动装载允许位

    TIM_Cmd(TIM4, ENABLE);   //启动定时器TIM4

    TIM_ITConfig(TIM4, TIM_IT_CC1|TIM_IT_CC2, ENABLE);

    NVIC_InitStructure.NVIC_IRQChannel = TIM4_IRQn ;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority=0 ;//抢占优先级0
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;      //子优先级0
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;         //IRQ通道使能
    NVIC_Init(&NVIC_InitStructure); //根据指定的参数初始化VIC寄存器

}
```

图 8 脉冲引脚初始化

这里是对脉冲引脚的初始化，主要是对 PB6、PB7 初始化为 PWM 翻转的模式，并且使能输出比较中断，计数值达到比较值时电平信号翻转，由高变低或由低变

高。主要是在下图中的中断中处理，先得到当前的计数值，再设置比较值为当前值加上我们设定的值。我们设定的值也是有一定的限制的为 0~900，步进电机的转度不能由 0 突变的很大，如果想让步进电机的转度达到很大，那就需要慢慢加速，虽然步进电机的转速上去了，但是转矩很小，轻轻一碰也会导致堵转或停止。

```
/**************************************************************************
函数功能：TIM4中断服务函数 步进电机调频PWM输出
入口参数：无
返回  值：无
**************************************************************************/
void TIM4_IRQHandler(void)
{
    float Capture1,Capture2;
    if(TIM_GetITStatus(TIM4,TIM_IT_CC1)!=RESET)
    {
        TIM_ClearITPendingBit(TIM4, TIM_IT_CC1 );//清除TIMx的中断待处理位
        Capture1 = TIM_GetCapture1(TIM4);
        TIM_SetCompare1(TIM4, Capture1 + Final_Moto2 );//设置TIMx自动重装载寄存器值
    }
    if(TIM_GetITStatus(TIM4,TIM_IT_CC2)!=RESET)
    {
        TIM_ClearITPendingBit(TIM4, TIM_IT_CC2 );//清除TIMx的中断待处理位
        Capture2 = TIM_GetCapture2(TIM4);
        TIM_SetCompare2(TIM4, Capture2 + Final_Moto1 );//设置TIMx自动重装载寄存器值
    }
}
```

图 9 输出比较中断

```
/**************************************************************************
函数功能：电机的正反转 和转速的调整
入口参数：
        moto：1控制电机A,2控制电机B
        speed：控制电机速度（0-900）
        direction：moto1：0为顺时针 1为逆时针
                   moto2：1为顺时针 0为逆时针
返回  值：无
**************************************************************************/

static int h,i,j,k;
void control(int moto,int speed,int direction)
{

    if(moto==1) {
//  if(Final_Moto1>1000-speed)
//    h++;
      Final_Moto1=1000-speed;
      PBout(15)=direction;
    }
    if(moto==2) {
//  if(Final_Moto2>1000-speed)
//    i++;
      Final_Moto2=1000-speed;
      PBout(13)=direction;
    }
}
```

图 10 设置电机转速和方向函数