《单片机应用系统设计》课程设计

设计报告书

题目名称：　直流电机控制系统

所在学院：　物理科学与技术学院

专　业　：　电子信息科学与技术

学生姓名：　张　伟　160901137

指导老师：　陈　磊

（2018年12月）

摘要

当今，自动化控制系统已经在各行各业得到了广泛的应用和发展，而直流电机驱动控制作为电器传动的主流在现代化生产中起着主要作用。长期以来，直流电动机因其转速调节比较灵活，方法简单，易于大范围平滑调速，控制性能好等特点，一直在传动领域占有统治地位。

本课程设计主要通过 PWM 调速实现直流电机的正转、反转、加速、减速、启停等操作，利用 PID 控制算法使系统更加快速和稳定。为实现系统的控制，采用了 STC15F2K60S2 增强型单片机作为整个控制系统的核心部分，配以 OLED 显示电机速度、AD 测量值、电机正反转等参数，实现系统的人机交互。不断采集霍尔编码器的脉冲数读取电机的转速，利用 PID 增量式方法快速在旋钮调解时趋向目标值。同时，通关过匿名上位机实时观测调节过程，或是超调，亦或是振荡都能及时的看出来。

通过外部中断、定时器中断、AD 中断操作，在方案实现的过程中，需要明确他们的优先级，防止发生冲突。这也是本系统设计的一个难题。

关键字 直流电机 单片机 PID PWM 编码器

目 录

第一部分 课程设计概述

1.1 课程设计的目的与任务

《单片机应用系统设计》课程设计是运用“应用系统设计”这门课程所学的应用系统设计的各方面知识，进行综合设计的一门课程。设计者要熟练掌握 Keil C 软件编程环境和 STC 单片机的片上资源，并通过查文献资料了解掌握相关外围器件的用法，综合运用进行设计。本课程是配合单片机原理和应用系统设计及模拟、数字电路，传感器原理等多门课程教学的一个重要实践环节，能帮助学生应用课堂学习到的知识，加强综合能力，提高系统设计水平。

1.2 课程设计题目

直流电机控制系统

1.3 设计功能要求

（1）焊接电机控制电路板，调试后电路功能正常； （30 分）

（2）可以通过 5 向按键左右控制电机正反转、侧向按键控制电机启动/停止；（10 分）

（3）通过旋纽电阻控制电机转速或角度（可二选一）；（20 分）

（4）相关电机转速或角度，转动方向信息在 LCD1602 或 OLED 上显示； （20 分）

（5）通过 PID 调节，通过旋纽控制转速或转动角度（可二选一），数据通过串口，在电脑上显示曲线（可借助匿名地面站工具），相应的在显示器上显示目标转速或角度。（15 分）

（6）其他。（5 分）

1.4 课程设计的内容与要求

（1）设计制作硬件电路；

（2）编写单片机系统软件；

（3）软硬件调试；

（4）写课程设计报告

1.5 实验仪器设备及器件

1）PC 机；

2）Keil C51 软件；

3）STC15F2K60S2 增强型单片机，TB6612 电机驱动芯片，直流电机，OLED 显示屏；

第二部分 设计方案工作原理

经过对本课程设计的分析，本直流电机控制系统将通过控制电机的转速完成相关设计要求。AD 采样判别五向按键，定时器控制系统工作时间，外部中断（In0P31 和 In1P32）读取编码器 A、B 相的脉冲数，单位时间内的总数即为电机的转速。测得的速度值和旋钮采样值通过 OLED 显示出来，通过 PID 算法实现速度控制，并在匿名上位机上实时显示目标值和当前值。不断调节比例系数和积分系数，是系统达到稳定性、快速性、准确性的基本要求。

2.1 预期实现目标定位

主要通过 PWM 调速实现直流电机的正转、反转、加速、减速、启停等操作，利用 PID 控制算法使系统更加快速和稳定。为实现系统的控制，采用了 STC15F2K60S2 增强型单片机作为整个控制系统的核心部分，配以 OLED 显示电机速度、AD 测量值、电机正反转等参数，实现系统的人机交互。不断采集霍尔编码器的脉冲数读取电机的转速，利用 PID 增量式方法快速在旋钮调解时趋向目标值。

2.2 技术方案分析

2.2.1 系统框图

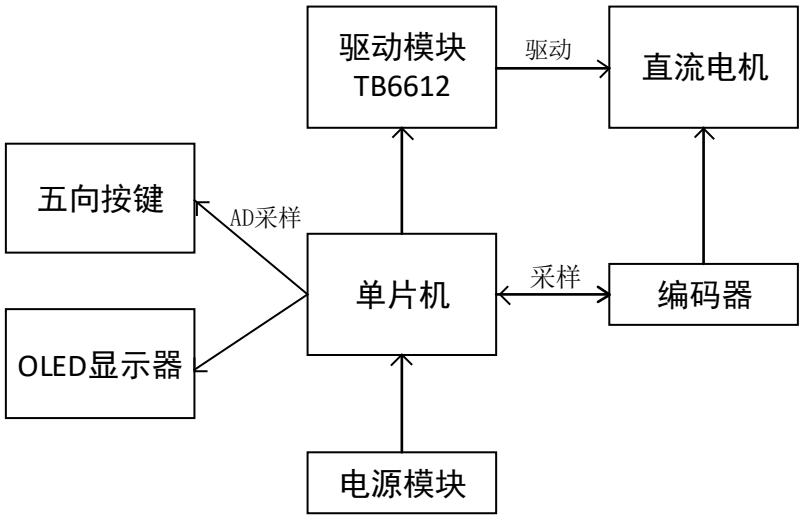

图 2.1 系统框图

2.2.2 电路工作原理

根据励磁方式不同， 直流电机分为自励和他励两种类型。 不同励磁方式的直流电机机械特性曲线有所不同。对于直流电机来说，认为机械特性方程式为：

n=UN/(Ke ϕ N)-(Rad+Ra)/(K eKt ϕ 2N)T=n-△n （公式 1-1）

式中 UN ， ϕ N ---------- 额定电枢电压、额定磁通量；

Ke ， Kt---与电机有关的常数；

Rad ， Ra-----电枢外加电阻、电枢内电阻；

n ， △n—理想空载转速、转速降。

分析公式 1-1 可得，当分别改变 UN 、ϕ N 和 Rad 时，可以得到不同的转速 n，从而实现对速度的调节。由于 ϕ =T，当改变励磁电流 If 时，可以改变磁通量 ϕ 的大小，从而达到变磁通调速的目的。 但由于励磁线圈发热和电动机磁饱和的限制，电动机的励磁电流 If 和磁通量 ϕ 只能在低于其额定值的范围内调节，故只能弱磁调速。而对于调节电枢外加电阻 Rad 时，会使机械特性变软，导致电机带负载能力减弱。

PWM 是通过控制固定电压的直流电源开关频率，从而改变负载两端的电压，进而达到控制要求的一种电压调整方法。PWM 可以应用在许多方面，如电机调速、温度控制、压力控制等。在 PWM 驱动控制的调整系统中，按一个固定的频率来接通和断开电源，并根据需要改变一个周期内“接通”和“断开”时间的长短。通过改变直流电机电枢上电压的“占空比”来改变平均电压的大小，从而控制电动机的转速。

2.2.3 控制算法原理

PID（比例（proportion）、积分（integral）、导数（derivative））通过线性组合构成控制量，用这一控制量对被控对象进行控制，这样的控制器称 PID 控制器。

一般情况下，这个反馈就是速度传感器返回给单片机当前电机的转速。简单的说，就是用这个反馈跟预设值进行比较，如果转速偏大，就减小电机两端的电压(电流)；相反，则增加电机两端的电压(电流) 。

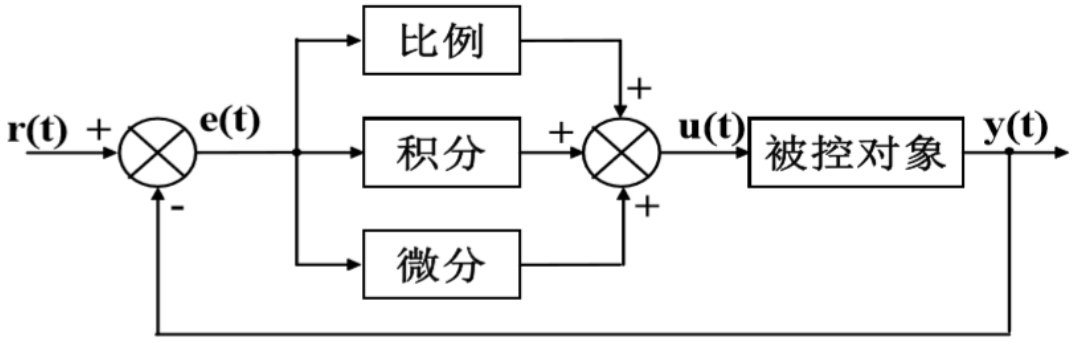

图 2.2 PID 原理图

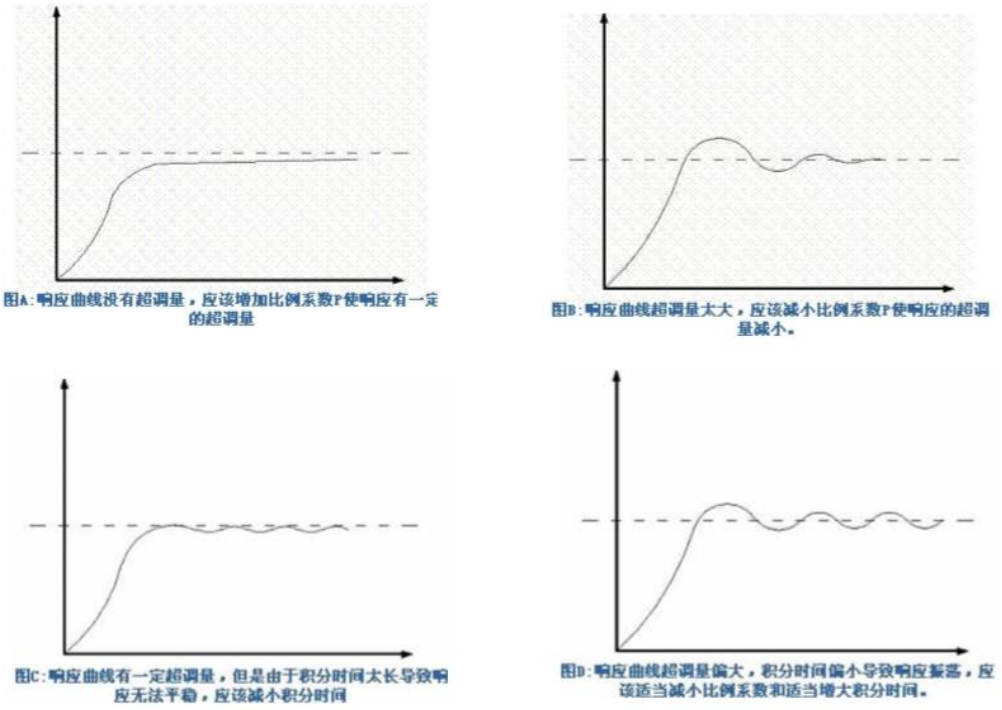

图 2.3 PID 调节分析

2.3 功能指标实现方法

2.3.1 实现方案分析

方案一：采用电阻网络或数字电位器调整电动机的分压，从而达到调速的目的。但是电阻网络智能实现有级调速， 而数字电阻的元器件价格比较昂贵。 更主要的问题在于一般电动机的电阻很小，但电流很大，分压不仅会降低效率，而且实现起来很困难。

方案二：采用继电器对电动机的开或关进行控制， 通过开关的切换对电机的速度进行调整。这个方案的优点是电路较为简单， 缺点是继电器的响应时间慢、机械结构容易损坏、寿命较短、可靠性不高。

方案三：采用驱动芯片 L298 驱动直流电机， L298 具有驱动能力强，外围电路简单等 优点。

综合各方面的因素，采用了**方案三**。

2.3.2 基本模块原理

1.驱动模块 TB6612

TB6612 是双驱动，也就是可以驱动两个电机。下面分别是控制两个电机的 IO 口，STBY 口接单片机的 IO 口清零电机全部停止，置 1 通过 AIN1 AIN2，BIN1，BIN2 来控制正反转。

驱动 1 路

PWMA　　接单片机的 PWM 口

VM　接 12V 以内电源

真值表：

VCC　接 5V 电源

AIN1 0　　0　　1

GND　接电源负极

AIN2 0　　1　　0

停止　正传　　反转

图 2.4 TB6612 驱动模块

2.编码器

霍尔器件是一种磁传感器，是半 导体材料制成的一种薄片，它是 一种磁敏感器件，当它处于磁场 中时，会产生电动势。在垂直磁 场平面方向上施加外磁场、在沿 平面上加外电场，则使电子在磁 场中运动，结果在器件的的两个 侧面之间产生霍尔电势，霍尔电 势的大小和外磁场以及电流大小 成正比。用它们可以检测磁场及 其变化，可在各种与磁场有关的 场合中使用。霍尔器件以霍尔效 应为其工作基础。

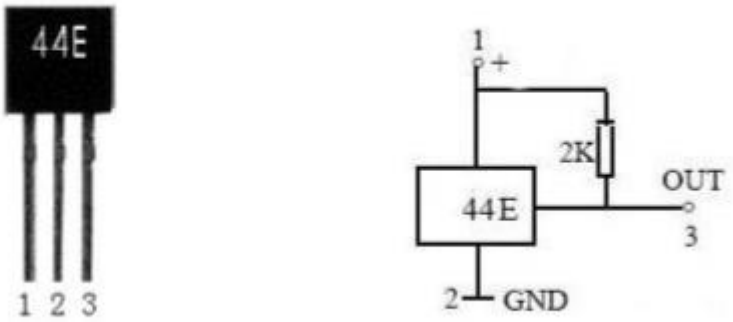

图 2.5 霍尔编码器

第三部分 核心部件电路设计

3.1 关键器件性能分析

本系统设计主要涉及电路驱动模块、电源模块、霍尔编码器测试模块，主要性能及原理详见第二部分中基本模块原理分析部分。

3.2 电路工作原理

主要涉及编码器的测速、电机驱动模块。

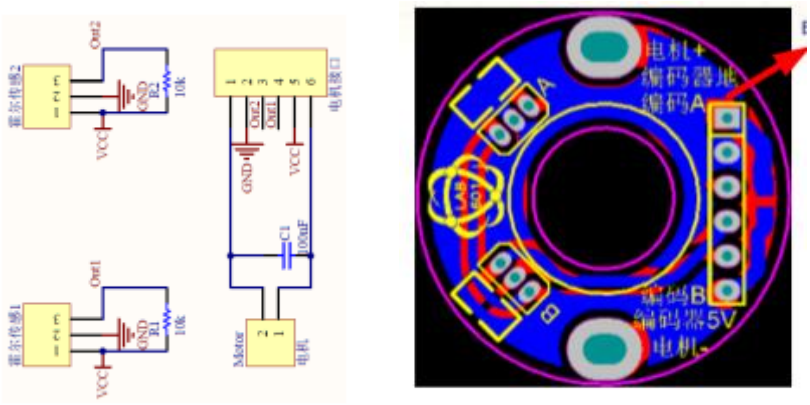

图 3.1 编码器

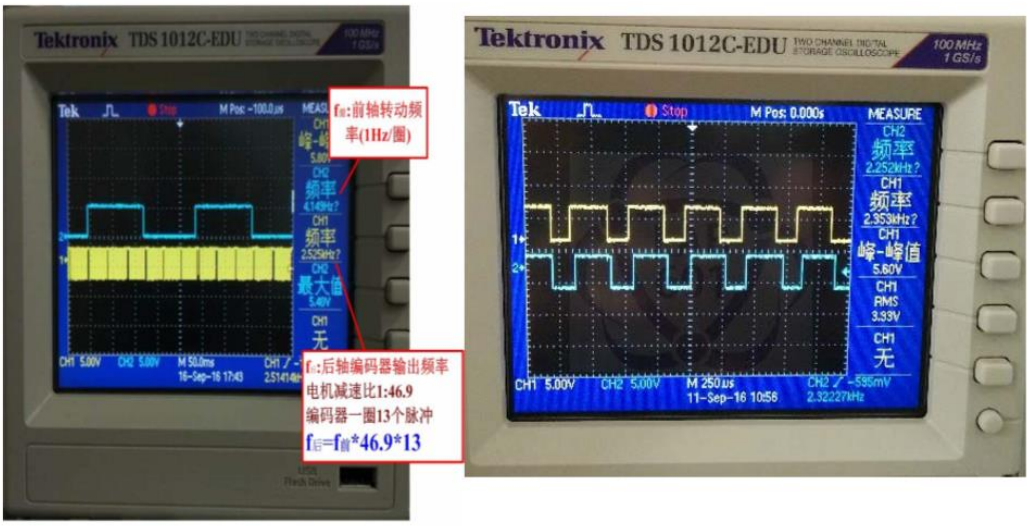

图 3.2 编码器测速原理示意

3.3 电路驱动接口说明

测试按键 P55，测试 LED P54，驱动 TB6612 INA0 P16、INA1 P15，PWMA P14，En P17。

第四部分 系统软件设计分析

4.1 系统总体工作流程

应用软件的编制和调试，使用 Keil 软件编程时，项目开发流程和其它软件开发项目的流程较为相似。

(1)创建一个项目，从器件库中选择目标器件，配置工具设置；

(2)用 C 语言或会变语言创建源程序；

(3)用项目管理器生成应用；

(4)修改源程序中的错误；

(5)测试，连接应用。

4.2 程序设计思路

利用 P3 口，编制程序输出一串脉冲，经放大后驱动直流电机，改变输出脉冲的电平的持续时间，达到使电机正转、反转、停止、加速、减速等目的。由软件编程从 P3.0/P3.1 管脚产生 PWM 信号，经驱动电路输出给电机，从而控制电机得电与失电。软件采用延时法进行设计。单片机上电后，系统进入准备状态。按动正转按钮后，根据 P3.0 为高电平时实现电机正转，P3.1 为高电平时实现电机反转。根据不同的加减速按钮，调整 P3.0/P3.1 输出高低电平时的有效值，进而控制电机的加减速。其主程序流程如图 2 所示。

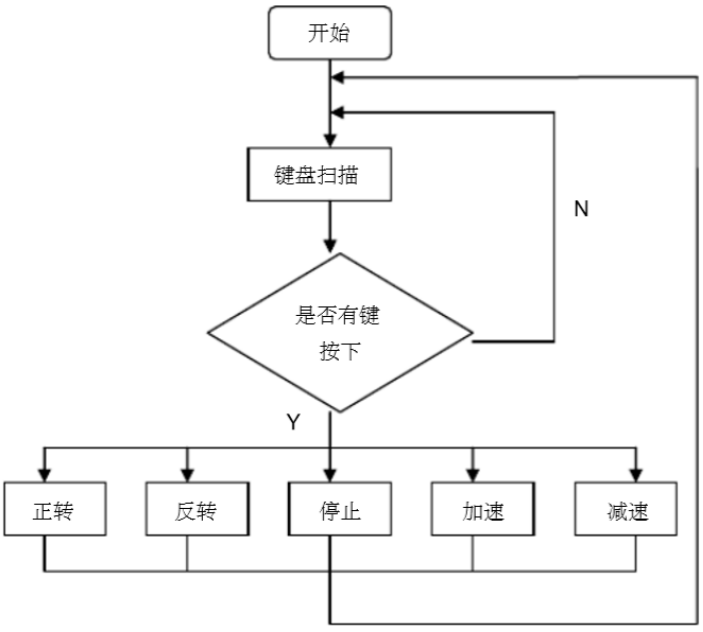

图 4.1 程序设计流程图

4.3 关键模块程序清单

4.3.1 编码器测速

```
void Timer0Interrupt() interrupt 1 using 2 //定时器 0 的中断
{
    T0Counter2ms++;
    if(T0Counter2ms>=2)
    {
        bFlag2ms=1;
        T0Counter2ms=0;
        T0Counter20ms++;
        ADC_CONTR = ADC_POWER | ADC_SPEEDHH | ADC_START | 0x00;//2ms 启动一
次 ADC
        if(T0Counter20ms >= 100)
        {
            aFlagAD = 0;
            ADC_CONTR = ADC_POWER | ADC_SPEEDHH | ADC_START | 0x01;//2ms 启
动一次 ADC
        }
        else{}
                if(T0Counter20ms >= 250){
                bFlag20ms=!bFlag20ms;
                T0Counter20ms=0;
              T0Counter1s++;
              //TestLed = !TestLed;
            if(T0Counter1s >= 2)
            {
                CSpeed = (APhra+BPhra) / 20;    //25ms 计算一次
            APhra = 0;BPhra = 0;
            }
        }
        else{}
    }
    else{}
}
```

4.3.2 五向按键检测

```
unsigned char Judge5DKeyNum(unsigned char ADCValue)
{
    static unsigned char xdata cFlag5D=0;//保存连接读入的键值
    static unsigned char xdata c5DkeyLastTime=0;//保存上次按键值
//  static unsigned char xdata cNumLastTime=6;//记录上次确认的键值
    static unsigned char xdata cKeyPressNum;
    unsigned char c5DkeyNow=7;
```

```
//    unsigned char c5DkeyNow=7;
//    unsigned char i=0;
//    //判断当前读入 ADC 对应的按键值

//    思路与单个按键相似
      //根据 AD 值得到当前键值
      if(ADCValue<=KeyThreshold[0]) c5DkeyNow=0;
      else if (ADCValue>KeyThreshold[0] && ADCValue<=KeyThreshold[1]) c5DkeyNow=1;
      else if (ADCValue>KeyThreshold[1] && ADCValue<=KeyThreshold[2]) c5DkeyNow=2;
      else if (ADCValue>KeyThreshold[2] && ADCValue<=KeyThreshold[3]) c5DkeyNow=3;
      else if (ADCValue>KeyThreshold[3] && ADCValue<=KeyThreshold[4]) c5DkeyNow=4;
      else if (ADCValue>KeyThreshold[4] && ADCValue<=KeyThreshold[5]) c5DkeyNow=5;
      else   c5DkeyNow=6;

      //记录按键
      if(c5DkeyNow==6)//抬起，记录一次
            cFlag5D=(cFlag5D<<1)&0;//左移记录 1 次
      else if(c5DkeyNow==c5DkeyLastTime)//AD 判断的键值与上次相同，
            cFlag5D=(cFlag5D<<1)|1;//左移记录 1 次
      else //特殊情况，本次非抬起，也与上次不同，基本不可能出现
            cFlag5D=(cFlag5D<<1)& 0;//左移记录 1 次
      c5DkeyLastTime=c5DkeyNow;//记录当前 AD 读的键值

      //判断键值
      if(cFlag5D==0xFF)//连续 8 次读入一样
            cKeyPressNum=c5DkeyNow;      //记录当前键值

   if(cFlag5D==0x00 && cKeyPressNum !=6 )//按键有效抬起，且前一次为有效按键
      {
            c5DkeyNow=cKeyPressNum;
            cKeyPressNum=0x06;
            return c5DkeyNow;
      }
      else
            return 0x06;
}
```

4.3.3 OLED 显示

```
void MENU_oLED(void)
{
      OLED_Clear();
      OLED_ShowCHinese(12,0,2);//直   //存放于数组  Hzk[][32]中
      OLED_ShowCHinese(30,0,3);//流
      OLED_ShowCHinese(48,0,4);//电
```

```
    OLED_ShowCHinese(66,0,5);//机
    OLED_ShowCHinese(84,0,6);//控
    OLED_ShowCHinese(102,0,7);//制
    OLED_ShowString(0,3,"ADVal:",48);
    OLED_ShowString(1,5,"Speed:        r/s",48);
    OLED_ShowString(0,7,"         12/29 10:42",12);
}

        //电子数字钟
        if(bFlag20ms == 0){
            tcount++;
            OLED_ShowString(0,7,"         12/29 10:42",12);
//              OLED_ShowNum(52,4,CSpeed,4,16);
            if(tcount>= 60) {
                tcount = 0;
                OLED_ShowString(0,7,"         12/29 10:43",12);
            }
        }
        else {
            if (tcount == 0) OLED_ShowString(0,7,"         12/29 10 43",12);
            else OLED_ShowString(0,7,"         12/29 10 42",12);
```

4.3.4 PID 控制

```
void IncPIDCalc1(unsigned int AimSpeed,unsigned int Current)
{
    static double ek = 0;//当前误差
    static double ek1 = 0,ek2 = 0; //
    static double Pwm = 0;
    float Kp = 1.312, Ki = 0.0001, Kd = 0; //上一次最好情况的参数    1.323
    //double Kp = 1.312, Ki = 0.0001, Kd = 0;
    ek = AimSpeed - Current; //增量计算
    Pwm += Kp * (ek - ek1) //E[k]项
                    +Ki * ek //E[k-1]项
                    +Kd * ((ek- ek1) - (ek1-ek2)); //E[k-2]项
    //存储误差，用于下次计算
    ek2 = ek1;
    ek1 = ek;
    //返回增量值
    if(Pwm > 255) Pwm = 250;
    if(Pwm < 0) Pwm = 5;
    //Test_Send_User(AimSpeed,Current,Pwm);     //AimSpeed -32
  PWM_Set(255-Pwm);
}
```

4.4 调试分析

4.4.1 总体说明

在调试时，由于子程序有很多，有时没法将每一个子模块都运行到，自然也无法及时发现其中的疏漏。 为了解决这个问题，更好地查找错误，我解决的方法是将各个模块分别进行调试。例如在调试某一模块时，先将其他模块用“ //”暂时屏蔽掉，直到各个子模块都健康运行后，再整体调试，这样我们更加容易找出错误，增加效率。软件调试需要不断的在单片机上执行看输出的结果，如果每次都在硬件上操作比较麻烦，因此我使用了 “Proteus”仿真软件，将我们的电路硬件搭建出来，在这个平台上调试软件，并且达到了比较好的效果。

根据实验结果，本设计基本完成了设计要求，系统能够实现正转、反转、加速、减速、停止能功能。但是由于我对数码管等显示模块掌握度不够，系统还不能显示出电机转速，如果可以再多给我一些时间，我一定能设法用 OLED 显示出电机的转速。

4.4.2 PID 算法调节分析

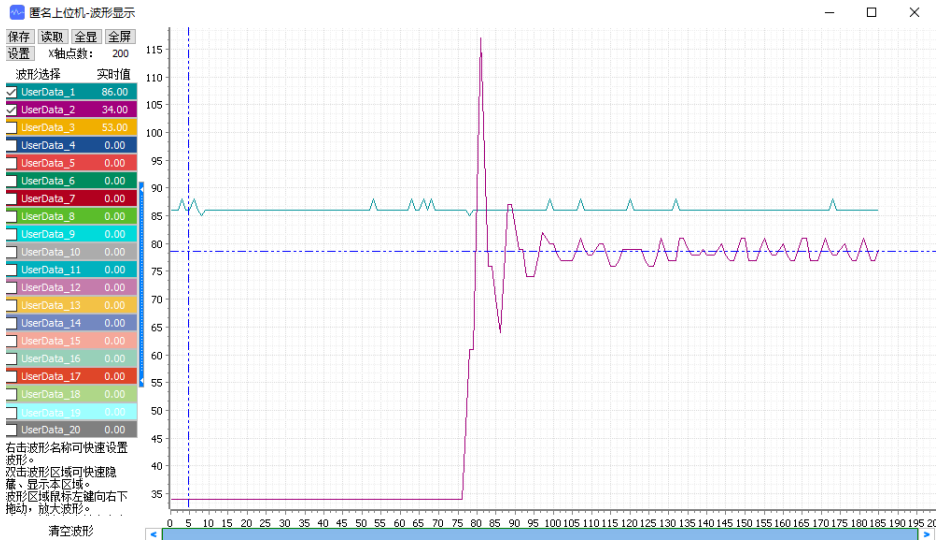

图 4.2 Kp = 1.312, Ki = 0, Kd = 0;

图 4.3 Kp = 1.312, Ki = 0.12, Kd = 0;

从上面两张实时波形图的结果，显而易见可以得出以下结论或者经验：

显然比例 P 越大时，电机转速回归到输入值的速度将更快，及调节灵敏度就越高。从而，加大 P 值，可以减少从非稳态到稳态的时间。但是同时也可能造成电机转速在预设值附近振荡的情形，所以又引入积分 I 解决此问题。

积分环节主要是用来消除静差，所谓静差就是系统稳定后输出值和设定值之间的差值，积分环节实际上就是偏差累计的过程，把累计的误差加到原有系统上以抵消系统造成的静差。

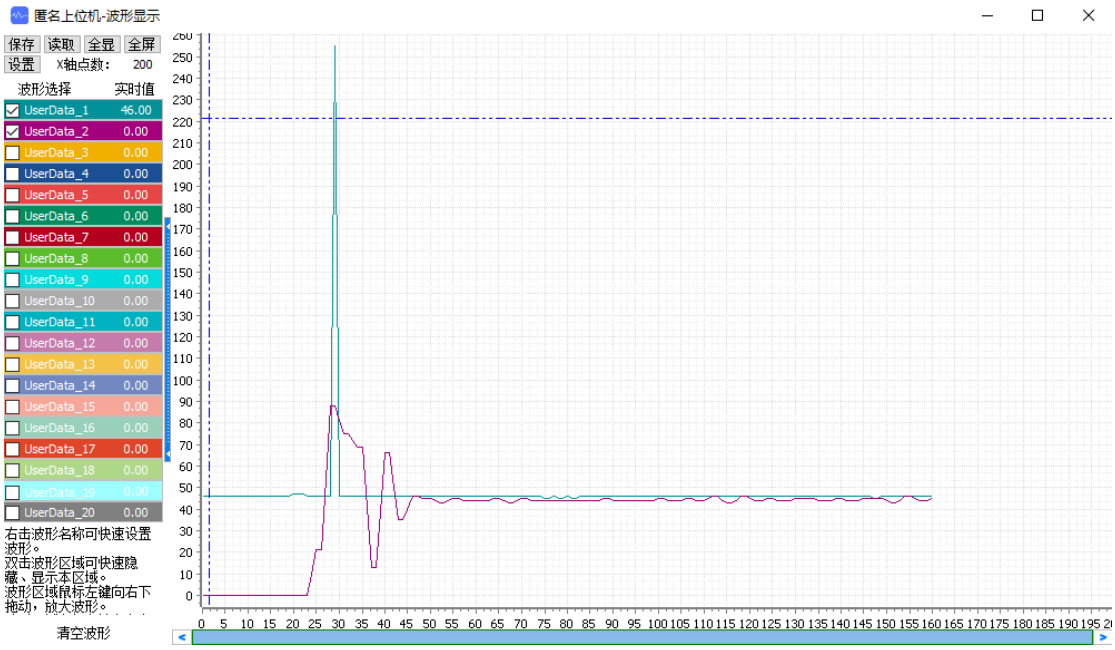

图 4.4 Kp = 1.312, Ki = 0.0001, Kd = 0;

第五部分 心得体会

通过本次课程设计，我从直流电机调速系统的设计与搭建中深深的体会到软件对于一个系统来说是多么重要。软件可以说是一个系统的灵魂 ，在工作中指导硬件按照指定的方案运行。对于刚学汇编不久的我来说 ，编制一个完整的系统软件可谓无任何经验可言。在设计过程中 ，我们主要学习体会了单个模块的搭建与编程 ，例如键盘子程序 ，中断子程序等等。在这个系统搭建过程中 ，不但要将这些子模块有机的结合在一起 ，还要让它们较好的协调起来 ，按照我们思路运行，可以说是比较困难的。

由于我经验不足，单片机编程是不能想当然的。我最容易犯的错误就是不经论证就去按照自己觉得可行的思路去进行，往往导致系统不能正常工作。例如显示间隔时间的问题，我本以为长短无所谓，但结果是如果间隔大于 10ms 就会产生闪烁感，导致没有静态的效果，而间隔太短的话必须在显示下一个之前，将前一个位选清除，否则就容易产生“串位”，导致显示的不正常；在键盘扫描子程序中，当有键按下做相应操作，必须当按键释放时才能继续扫描，否则将导致一次按下执行多次的错误情况… …虽然遇到了许多困难，但是在老师的帮助下，我还是完成了这次的课程设计。通过本次课程设计， 我进一步了解了系统搭建的过程和系统软件编程的步骤，为今后的学习打下良好的基础。

在这里我要感谢我的指导老师陈老师。老师工作很忙，但还是在我做课程设计的时间里一直关心我的进展，从设计方案的确定和修改，仿真的检查，及后来的详细设计等过程中都给了我很大的支持和关注。

本次课程设计让我把理论应用到了实践， 同时通过课程设计，也加深了我对专业理论知识的理解和掌握。在解决问题的过程中， 我查阅了大量专业书籍，获得了许多专业知识，拓展了视野，提高了我的理论水平和实际的动手能力，并让我学会了解决问题的方法，激发了我的探索精神。这样的课程设计是很好的锻炼机会，课程设计使我深入的了解到了实践能力对于工科学生的重要性， 增强了我们的实践动手能 力。

第六部分 附录

Ⅰ参考文献

[1] 王鉴光 ．电动机控制系统 ．北京：北京机械工业出版社 ,1994
[2] 王小明 ．电动机的单片机控制 ．北京：北京航空航天大学出版社 ,2002
[3] 张垛. 直流无刷电动机原理及应用 ．北京：北京机械工业出版社 ,1996

Ⅱ电路原理图

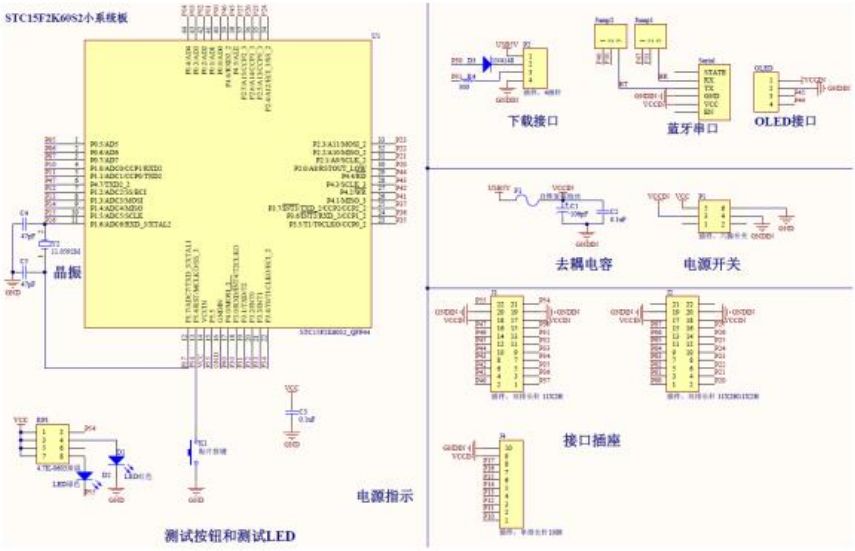

图 i-1STC15F2K60S2 最小系统版

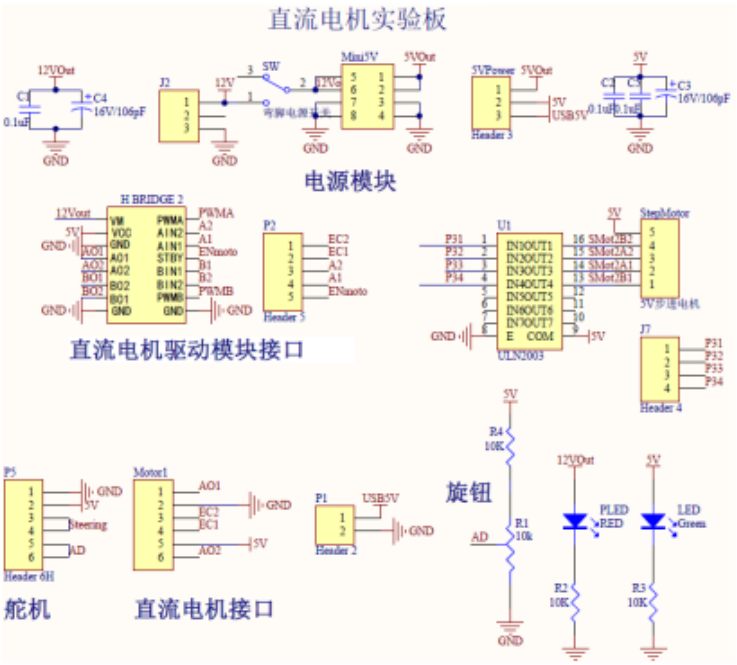

图 i-2 直流电机实验板

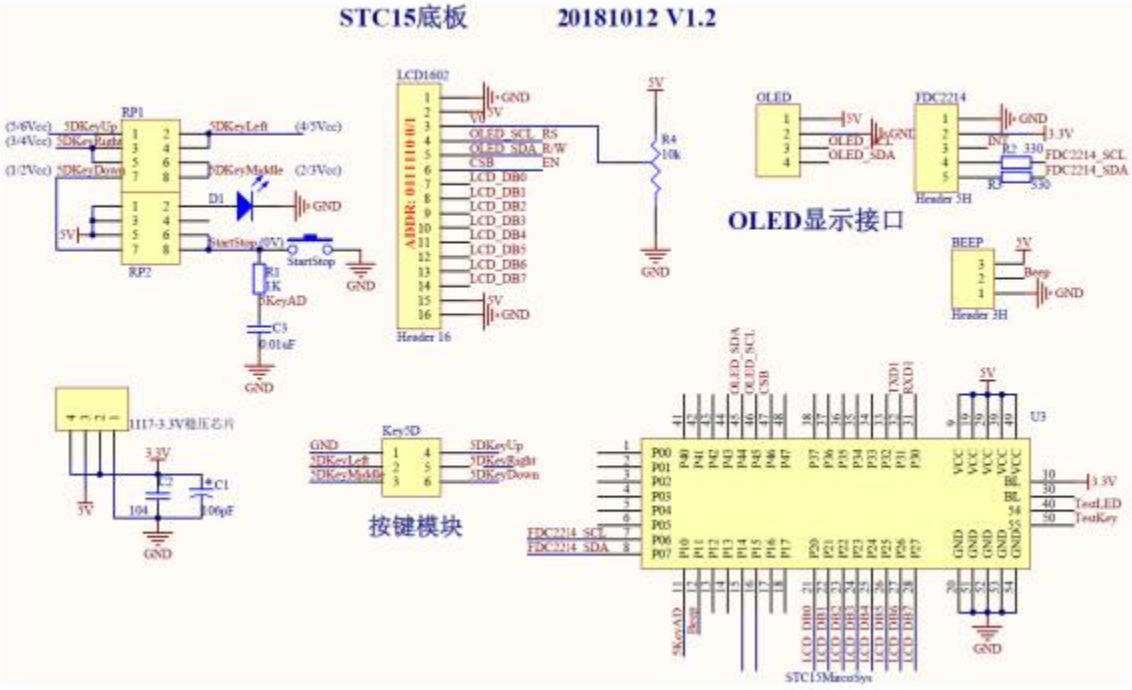

图 i-3 STC 底板原理图

III 源代码

```
void main()
{
    extern bit bS1Rec;//串口接收到数据标志
    extern bit bFlag2ms,bFlag20ms;//2ms 标志
    extern bit bFlagAD;//AD 转换标志
    extern bit aFlagAD;//AD 转换标志
    extern unsigned char cS1Rec;//串口接收到字符
    extern unsigned int cADCResult;//AD 高 8 位
  extern unsigned int AngADResult;     //用于读取旋钮的角度
    extern unsigned char ch;          //ADC 通道号
    extern bit aFlagAD;//AD 转换标志
//    extern unsigned int xdata CSpeed;   //电机当前的速度  每 0.5 秒测一次

    unsigned int xdata aimspeed=0, last_speed = 0;
    extern unsigned int xdata CSpeed;   //电机当前的速度  每 0.5 秒测一次

    unsigned char xdata cKeyPressNumber=0x06;
    unsigned char* xdata cindexTemp=0;
    unsigned int xdata current_speed = 0;
    unsigned int ucount = 0,tcount = 0;//波形打印
    bit motorflag = 1;
    bit flag = 0;
    bFlag2ms=0;bFlag20ms=0;
```

```
    GDD = 0;

    //Beep = 0;
    MyGPIO_Inilize();//GPIO
    Uart1Init();//串口 1 初始化
    Timer0Init();//定时器 0 初始化
    Int0Init(); //外部中断初始化
    ADCInit(); //ADC 初始化
    Motor_Init(); //电机初始化
    OLED_Init();  //初始化 OLED
    MENU_oLED();    //OLED 菜单显示
    PWM_Init(0x00);
    //IncPIDInit();

    while(1)
    {
        //读取按键值
        if(bFlagAD==1) //2m 读取一次按键 AD 值（五向按键和侧向按键）
        {
            bFlagAD=0;
            ucount++;
            cKeyPressNumber=Judge5DKeyNum(cADCResult);//读取按键值
        }
        else{}

        //电子数字钟
        if(bFlag20ms == 0){
            tcount++;
            OLED_ShowString(0,7,"         12/29 10:42",12);
//          OLED_ShowNum(52,4,CSpeed,4,16);
            if(tcount>= 60) {
                tcount = 0;
                OLED_ShowString(0,7,"         12/29 10:43",12);
            }
        }
        else {
            if (tcount == 0) OLED_ShowString(0,7,"         12/29 10 43",12);
            else OLED_ShowString(0,7,"         12/29 10 42",12);
        }

        //实时显示旋钮 AD 值
        if(aFlagAD == 0)
        {
            ET0 = 0;ch = 1; //关定时器
```

```
        ADC_CONTR = ADC_POWER | ADC_SPEEDLL | ADC_START | ch;
        while(aFlagAD == 0) ;
                OLED_ShowNum(52,2,AngADResult,3,16);
            //PWM_Set(255-aimspeed);
            //Test_Send_User(aimspeed,CSpeed,00);
        //修改：根据读到的 AD 对应的标度作为速度的目标值
        ch = 0;ET0 = 1;
    }
    else{       }

    //每 50ms 调节一次 PID
    if(ucount >= 10){    //20ms
        ucount = 0;
        current_speed = CSpeed;
        current_speed = (last_speed + CSpeed)/2;
        aimspeed = 1.3813 * AngADResult + 47.978; //线性标度目标值
        OLED_ShowNum(52,4,current_speed*3/2,4,16);
        IncPIDCalc1(aimspeed,current_speed);
        Test_Send_User(AngADResult,current_speed,00);      //AimSpeed -32
        if(!(abs(current_speed-aimspeed) <= 5))
        {
            TestLed = !TestLed; //directing the PID is runnning
        }
        else{
            Beep = !Beep;
            //DelayNms(1000);
            Beep = !Beep;
            //DelayNms(1000);
        }
        last_speed = CSpeed;
}
    else{}

    switch(cKeyPressNumber) //根据按键执行
    {
        case 0: //KeyStartStop  直流电机启动与停止
        {
            if(!flag)
            {
                flag = 1;
                Motor_Start();
            }
            else{
                flag = 0;
```

```
            Motor_Stop();
        }
        cKeyPressNumber=0x06;break;//串口发送 00;
}

case 1://KeyDown  向下键每次减 5%
{
          TestLed = !TestLed;
          //Motor_dec();
            //S1SendData(0x01);
            cKeyPressNumber=0x06;break;//串口发送 01;
}

case 2:     //KeyMiddle
{
            //S1SendData(0x02);
        cKeyPressNumber=0x06;break;//串口发送 02;
}

case 3: //KeyRight  直流电机正转
{
            Motor_Turn(0x01); //1 正转
          OLED_ShowCHinese(110,2,8);//正
            //S1SendData(0x03);
            cKeyPressNumber=0x06;break;//串口发送 03;
}

case 4: //KeyLeft    直流电机反转
{
            Motor_Turn(0x00); //0  反转
        OLED_ShowCHinese(110,2,9);//反
            //S1SendData(0x04);
            cKeyPressNumber=0x06;break;//串口发送 04;
}

case 5: //KeyUp      向上键每次增 5%
{
            Motor_add();
            //S1SendData(0x05);
            cKeyPressNumber=0x06;break;//串口发送 05;
}

default:// NoneKey
          //S1SendData(0x06);
```

```
                        break;
                }
        }
}


void Int0Init(void)        //外部中断 0\1（下降沿）
{
        IT0 = 0;       //设置 INT0 的中断类型  (1:仅下降沿  0:上升沿和下降沿)
        EX0 = 1;       //使能 INT0 中断
      IT1 = 0;       //设置 INT1 的中断类型  (1:仅下降沿  0:上升沿和下降沿)
      EX1 = 1;       //使能 INT1 中断
        PX0 = 0;
        PX1 = 0;
      EA = 1;
}

//中断服务程序
void exint0() interrupt 0      //INT0 中断入口
{
   APhra++;
      //TestLed = !TestLed;         //将测试口取反
}

void exint1() interrupt 2 //INT1 中断入口
{
   BPhra++;
      //TestLed = !TestLed;         //将测试口取反
}


bit KeyPress(bit KeyIO)
{
      cJudgeKey=(cJudgeKey<<1) | KeyIO; //  判断值左移一位 ，并将当前扫描值入最低
      if(cJudgeKey==0x00)//按键按下，状态稳定。KeyIO 按下为 0，抬起为 1
      {
            TestKeyLastTime=KeyIO;//记录状态
      }
      else if((cJudgeKey==0xFF) && (TestKeyLastTime!=KeyIO))//按键从按下到抬起，稳定
      {
            TestKeyLastTime=KeyIO;
            return 1;
      }
      else
```

```
    {

    }
    return 0;
}

unsigned char Judge5DKeyNum(unsigned char ADCValue)
{
    static unsigned char xdata cFlag5D=0;//保存连接读入的键值
    static unsigned char xdata c5DkeyLastTime=0;//保存上次按键值
//  static unsigned char xdata cNumLastTime=6;//记录上次确认的键值
    static unsigned char xdata cKeyPressNum;
    unsigned char c5DkeyNow=7;



//  unsigned char c5DkeyNow=7;
//  unsigned char i=0;
//  //判断当前读入 ADC 对应的按键值

//  思路与单个按键相似
    //根据 AD 值得到当前键值
    if(ADCValue<=KeyThreshold[0]) c5DkeyNow=0;
    else if (ADCValue>KeyThreshold[0] && ADCValue<=KeyThreshold[1]) c5DkeyNow=1;
    else if (ADCValue>KeyThreshold[1] && ADCValue<=KeyThreshold[2]) c5DkeyNow=2;
    else if (ADCValue>KeyThreshold[2] && ADCValue<=KeyThreshold[3]) c5DkeyNow=3;
    else if (ADCValue>KeyThreshold[3] && ADCValue<=KeyThreshold[4]) c5DkeyNow=4;
    else if (ADCValue>KeyThreshold[4] && ADCValue<=KeyThreshold[5]) c5DkeyNow=5;
    else   c5DkeyNow=6;

    //记录按键
    if(c5DkeyNow==6)//抬起，记录一次
        cFlag5D=(cFlag5D<<1)&0;//左移记录 1 次
    else if(c5DkeyNow==c5DkeyLastTime)//AD 判断的键值与上次相同，
        cFlag5D=(cFlag5D<<1)|1;//左移记录 1 次
    else //特殊情况，本次非抬起，也与上次不同，基本不可能出现
        cFlag5D=(cFlag5D<<1)& 0;//左移记录 1 次

    c5DkeyLastTime=c5DkeyNow;//记录当前 AD 读的键值

    //判断键值
    if(cFlag5D==0xFF)//连续 8 次读入一样
        cKeyPressNum=c5DkeyNow;      //记录当前键值

  if(cFlag5D==0x00 && cKeyPressNum !=6 )//按键有效抬起，且前一次为有效按键
```

```
    {
        c5DkeyNow=cKeyPressNum;
        cKeyPressNum=0x06;
        return c5DkeyNow;
    }
    else
        return 0x06;
}

unsigned char* Hex2ASCII(long int long28Value)//8 位值转化为 ASCII
{
    unsigned char xdata * cindexTempHex=0;
    char xdata i=0;

    for(i=7;i>=0;i=i-2)//高位在前
    {
        cindexTempHex[i]=long28Value;//利用默认数据类型转换，char 为 8 位，取出 lont int
的低 8 位
        cindexTempHex[i-1]=cindexTempHex[i]>>4;//取出 8 位中高 4 位
        cindexTempHex[i]=cindexTempHex[i]-(cindexTempHex[i-1]<<4);//取出 8 位中的低 4
位
        long28Value=long28Value>>8;//低 8 位处理完毕，右移
    }
//  S1SendData(0xAA);
    for(i=0;i<=7;i++)
    {

//      S1SendData(cindexTempHex[i]);
        if(cindexTempHex[i]<=9) cindexTempHex[i]+=0x30;//小于 9 转成 ASCII
        else cindexTempHex[i]=cindexTempHex[i]+55;//大于 9 的数转成 ASCII
//      S1SendData(cindexTempHex[i]);
    }
    cindexTempHex[8]=0;//数组后加一个结束符
    return cindexTempHex;
}

void MyGPIO_Inilize()
{
    GPIO_Inilize(GPIO_P0,GPIO_PullUp);//IO 初始化//上拉准双向口
    GPIO_Inilize(GPIO_P1,GPIO_PullUp);//IO 初始化//上拉准双向口
    GPIO_Inilize(GPIO_P2,GPIO_PullUp);//IO 初始化//上拉准双向口
    GPIO_Inilize(GPIO_P3,GPIO_PullUp);//IO 初始化//上拉准双向口
    GPIO_Inilize(GPIO_P4,GPIO_PullUp);//IO 初始化//上拉准双向口
    GPIO_Inilize(GPIO_P5,GPIO_PullUp);//IO 初始化//上拉准双向口
```

```
    GPIO_Inilize(GPIO_P5.4,GPIO_HighZ);//测试按键口浮空输入
    DelayNms(1);//空调用，避免占用低 128 字节的 Data 空间
}

void Motor_Init(void)
{
    A1 = 0;
    A2 = 0;
    En = 0;
}

void Motor_Start(void)
{
    En = 1;
    A1 = 0;
    A2 = 1;
}

void Motor_Stop(void)
{
    A1 = 0;
    A2 = 0;
}
void Motor_Turn(bit n)
{
    En = 1;
    if(n){      A1 = 0;A2 = 0;      DelayNms(200);A1 = 1; A2 = 0;}
    else{ A1 = 0;A2 = 0; DelayNms(200); A1 = 0; A2 = 1;}
}
```