LC3100_9 软 件 源 程 序

```
    #include "Speed.h"

/*
**********************************************************************
 * 函数原型：void Encoder_Init(void)
 * 功      能：编码器初始化
**********************************************************************
*/
void Encoder_Init(void)
{
    HAL_TIM_IC_Start_IT(&htim1, TIM_CHANNEL_1);//motor1  输入捕获
}

/*
**********************************************************************
 * 函数原型：void Check_Speed(float dT)
 * 功      能：检测速度是否停止-0.05s
**********************************************************************
*/
void Check_Speed(float dT)
{
    Speed_Cnt++;//每 50ms 进入
    if(Speed_Cnt >= 10)//0.5s 发现没出发输入捕获
    {
        Rel_Speed = 0;//将速度清零
        Speed_Cnt = 0;//计数清零
    }
}

/*
**********************************************************************
 * 函数原型：void TIM1CaptureChannel1Callback(void)
 * 功      能：Tim1 通道 1 的输入捕获回调函数
**********************************************************************
*/
uint32_t Capture, Capture1, Capture2;
uint32_t rel;
void TIM1CaptureChannel1Callback(void)
{
    Capture1 = __HAL_TIM_GET_COMPARE(&htim1, TIM_CHANNEL_1);//读取编码器的
值
    if(Capture1 > Capture2)//假如读到的值大于上一次的值
        Capture = Capture1 - Capture2;//得出当前的值
    else
        Capture = Capture1 + (0xFFFF - Capture2);//如果小于上次的值，0xFFFF-上一次的值
    if(Capture < 100)//过滤小于 100 的
        return;
    rel = 60000000 / (Capture * 12);//用 1S/编码器的值*一圈的脉冲
    Capture2 = Capture1;//将读到的值赋值给上一次的值
    if((rel - Rel_Speed < 1000 && rel - Rel_Speed > 0)  || (Rel_Speed - rel < 1000 &&
```

```
Rel_Speed - rel > 0))
    {
        Rel_Speed = rel;
    }
    Speed_Cnt = 0;//将检测速度的时间清零
}

/*
**********************************************************************
 * 函数原型：void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
 * 功    能：TIM_IC 回调函数
**********************************************************************
*/
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
    if(htim->Instance == TIM1)
    {
        if(htim->Channel==HAL_TIM_ACTIVE_CHANNEL_1)
        {
            TIM1CaptureChannel1Callback();
        }
    }
}
#include "Drv_Key.h"

/**********全局变量声明******/
uint8_t Key_Status;//按键按下标志

/**********局部变量声明******/
float Key_Cnt1,Key_Cnt2,Key_Cnt3,Key_Cnt4,Key_Cnt5,Key_Cnt6;//按下时间
uint8_t Key_Flag1,Key_Flag2,Key_Flag3,Key_Flag4,Key_Flag5,Key_Flag6;//按键按下标志
uint8_t LongPress1,LongPress2,LongPress3,LongPress4,LongPress5,LongPress6;//按键长按标志

/*
**********************************************************************
 * 函数原型：void Check_Press(float dT)
 * 功    能：检测按键按下状态-500ms
**********************************************************************
*/
void Check_Press(float dT)
{
    if(Key_Status)//按键按下
        Key_Status -= dT;//倒计时
}

/*
**********************************************************************
 * 函数原型：void Key_Scan(float dT)
 * 功    能：矩阵按键扫描
**********************************************************************
```

```
*/
void Key_Scan(float dT)
{
    ROW1_L;
    ROW2_H;
    ROW3_H;
    /*************************************P                                    键
************************************/
    if(HAL_GPIO_ReadPin(COL1_GPIO_Port,COL1_Pin) == 0)//按下按键
    {
        if(sys.Run_Status)//如果在显示离心力的模式下都不能操作
            return;
        if(LongPress1 == 0)//没有长按过
        {
            Key_Cnt1 += dT;//按下时间++
            Key_Flag1 = 1;//按键按下标志置一
        }
    }
    if(Key_Flag1 == 1)//按键被按下
    {
        if(HAL_GPIO_ReadPin(COL1_GPIO_Port,COL1_Pin) == 1)//抬起按键
        {
            if(Key_Cnt1 < 5)//小于 1.5S 是单击
            {
                if(Show_Circle == 0)
                {
                    Show_Circle = 1;//显示外框和 P-几模式
                    PMode_Option = 1;//记忆模式 1
                    Param_Read();//读取参数
                    SetOK_Flag = 1;//设置参数
                    sys.SetMode_Option = 1;//进入设置 P 值模式
                    Twinkle_Time = 6;//闪烁时间 6S
                }
                else
                {
                    Show_Circle = 0;//不显示外框和 P-几模式
                    PMode_Option = 0;//记忆模式 0
                    Param_Read();//读取参数
                    sys.SetMode_Option = 0;//模式清零
                    SetOK_Flag = 1;//设置参数
                }
                Beep_Time = 0.1;//蜂鸣器响 0.1S
            }
            Key_Flag1 = 0;//按键事件结束，等待下一次按下
            LongPress1 = 0;//长按标志清零
            Key_Cnt1 = 0;//按钮计数清零
        }
        if(Key_Cnt1 > 5 && Key_Cnt1 < 6)//按键时间大于 1.5S 小于 3S 表示长按
        {
            if(LongPress1 == 0)//如果没有一直一直长按着
```

```
            {
                if(Speed_ModeFlag)//假如在离心力的模式下
                {
                    Speed_ModeFlag = 0;//长按后推出
                    sys.SetMode_Option = 0;
                }
                else
                {
                    Speed_ModeFlag = 1;//显示 P-几的离心率模块
                    Param_Read();
                    sys.SetMode_Option = 5;
                }
                Beep_Time = 0.1;//蜂鸣器响 0.1S
                Twinkle_Time = 6;//闪烁时间 6S
                LongPress1 = 1;//长按标志置一
            }
        }
    }
    /************************************** 减 键
**************************************/
    if(HAL_GPIO_ReadPin(COL2_GPIO_Port,COL2_Pin )== 0)//按下按键
    {
        if(sys.Run_Status)
            return;
        Key_Cnt2 += dT;//按下时间++
        Key_Flag2 = 1;//按键按下标志置一
    }
    if(Key_Flag2 == 1)//按键被按下
    {
        if(HAL_GPIO_ReadPin(COL2_GPIO_Port,COL2_Pin) == 1)//抬起按键
        {
            if(Key_Cnt2 < 1.5)//小于 1.5S 是单击
            {
                if(sys.SetMode_Option == 2)//设置时间
                {
                    if(Time_Unit == 0)//时间状态是秒为单位时
                        Set_Time -= 10;//时间减 10S
                    else
                        Set_Time -= 60;//时间减 60S
                    if(Set_Time < 30)//设置时间小于 30S 时
                        Set_Time = 30;//设置时间等于 30S
                }
                else if(sys.SetMode_Option == 3)//设置速度
                {
                    if(Speed_Unit)
                    {
                        Set_Speed -= 100;//离心率加 100
                        if(Set_Speed < 100)//速度小于 100 时
                            Set_Speed = 100;//速度小于 100 时
                    }
```

```
                    else
                    {
                        Set_Speed -= 100;//速度减 100
                        if(Set_Speed < 500)//速度小于 500 时
                            Set_Speed = 500;//速度等于 500
                    }
                }
                else if(sys.SetMode_Option == 4)//设置温度
                {
                    Safe_Set_Temp -= 10;//温度减 1 度
                    if(Safe_Set_Temp  <  (Safe_Rel_Temp+50))//设定安全温度大于实际安
全温度+5 度时
                        Safe_Set_Temp  =  Safe_Rel_Temp+50;//设定安全温度等于实际安
全温度+5 度
                }
                else if(sys.SetMode_Option == 5)//设置离心率模块
                {
                    Speed_Mode -= 1;//模块类型减一
                    if(Speed_Mode < 1)//模块类型小于一
                        Speed_Mode = 1;//模块类型等于一
                }
                if(Show_Circle == 1 && sys.SetMode_Option == 1)
                {
                    PMode_Option--;
                    if(PMode_Option < 1)
                    {
                        PMode_Option = 1;
                    }
                    Param_Read();
                }
                Key_Status = 2;//设置时 2S 不闪烁
                Beep_Time = 0.1;//蜂鸣器响 0.1S
                Twinkle_Time = 6;//闪烁时间 6S
            }
            Key_Flag2 = 0;//按键事件结束，等待下一次按下
            Key_Cnt2 = 0;//按钮计数清零
        }
        if(Key_Cnt2 > 1.9 && Key_Cnt2 < 2.1)//按键时间大于 1.9S 小于 2.1S 表示长按
        {
            if(sys.SetMode_Option == 2)//设置时间
            {
                if(Time_Unit == 0)//时间状态是秒为单位时
                        Set_Time -= 60;//时间减 60S
                    else
                        Set_Time -= 600;//时间减 600S
                if(Set_Time < 30)//设置时间小于 30S 时
                    Set_Time = 30;//设置时间等于 30S
            }
            else if(sys.SetMode_Option == 3)//设置速度
            {
```

```
                    if(Speed_Unit)
                    {
                        Set_Speed -= 1000;//离心率加 1000
                        if(Set_Speed < 100)//速度小于 100 时
                            Set_Speed = 100;//速度小于 100 时
                    }
                    else
                    {
                        Set_Speed -= 1000;//速度减 1000
                        if(Set_Speed < 500)//速度小于 500 时
                            Set_Speed = 500;//速度等于 500
                    }
                }
                else if(sys.SetMode_Option == 4)//设置温度
                {
                    Safe_Set_Temp -= 100;//温度加 10 度
                    if(Safe_Set_Temp  <  (Safe_Rel_Temp+50))//设定安全温度大于实际安全温
度+5 度时
                            Safe_Set_Temp  =  Safe_Rel_Temp+50;//设定安全温度等于实际安
全温度+5 度
                }
                else if(sys.SetMode_Option == 5)//设置离心率模块
                {
                    Speed_Mode -= 1;//模块类型减一
                    if(Speed_Mode < 1)//模块类型小于一
                        Speed_Mode = 1;//模块类型等于一
                }
                Key_Status = 2;//设置时 2S 不闪烁
                Key_Flag2 = 0;//按键事件结束，等待下一次按下
                Key_Cnt2 = 1.5;//按钮计数清零
                Beep_Time = 0.1;//蜂鸣器响 0.1S
                Twinkle_Time = 6;//闪烁时间 6S
            }
        }
        ROW1_H;
        ROW2_L;
        ROW3_H;
        /*************************************MENU                                    键
**************************************/
        if(HAL_GPIO_ReadPin(COL1_GPIO_Port,COL1_Pin) == 0)//按下按键
        {
            if(sys.Run_Status == 1)
                return;
            if(LongPress3 == 0)//没有长按过
            {
                Key_Cnt3 += dT;//按下时间++
                Key_Flag3 = 1;//按键按下标志置一
            }
        }
        if(Key_Flag3 == 1)//按键被按下
```

```
    {
        if(HAL_GPIO_ReadPin(COL1_GPIO_Port,COL1_Pin) == 1)//抬起按键
        {
            if(Key_Cnt3 < 1.5)/*单击*///小于 1.5S 是单击
            {
                if(Show_Circle == 0)
                {
                    sys.SetMode_Option++;//设置模式++
                    if(sys.SetMode_Option > 4)//退出设置
                        sys.SetMode_Option = 0;//清零
                    if(sys.SetMode_Option == 1)
                        sys.SetMode_Option = 2;
                }
                else
                {
                    sys.SetMode_Option++;//设置模式++
                    if(sys.SetMode_Option > 4)//退出设置
                        sys.SetMode_Option = 0;//清零
                }
                Beep_Time = 0.1;//蜂鸣器响 0.1S
                Twinkle_Time = 6;//闪烁时间 6S
            }
            Key_Flag3 = 0;//按键事件结束，等待下一次按下
            LongPress3 = 0;//长按标志清零
            Key_Cnt3 = 0;//按钮计数清零
        }
        if(Key_Cnt3 > 1.5 && Key_Cnt3 < 3)//按键时间大于 1.5S 小于 3S 表示长按
        {
            if(LongPress3 == 0)/*长按*///如果没有一直一直长按着
            {
                if(Speed_Unit)//假如在离心力的模式下
                {
                    Speed_Unit = 0;//显示速度单位
                    Param_Read();
                }
                else
                {
                    Speed_Unit = 1;//显示离心力单位
                    Param_Read();
                }
                sys.SetMode_Option = 0;
                Beep_Time = 0.1;//蜂鸣器响 0.1S
                Twinkle_Time = 6;//闪烁时间 6S
                LongPress3 = 1;//长按标志置一
            }
        }
    }
    /**************************************Start                                    键
*************************************/
    if(HAL_GPIO_ReadPin(COL2_GPIO_Port,COL2_Pin )== 0)//按下按键
```

```
    {
        if(LongPress4 == 0)//没有长按过
        {
            Key_Cnt4 += dT;//按下时间++
            Key_Flag4 = 1;//按键按下标志置一
        }
    }
    if(Key_Flag4 == 1)//按键被按下
    {
        if(HAL_GPIO_ReadPin(COL2_GPIO_Port,COL2_Pin) == 1)//抬起按键
        {
            if(Key_Cnt4 < 1.5)//小于 1.5S 是单击
            {
                if((HAL_GPIO_ReadPin(UC_IN1_GPIO_Port,UC_IN1_Pin)==
1)&&(HAL_GPIO_ReadPin(UC_IN2_GPIO_Port,UC_IN2_Pin)== 1))//电磁锁 1 和 2 闭合时
                {
                    if(sys.Run_Status == 0)
                    {
                        if(Show_Circle == 1)
                            Circle_Run = 1;
                        Speed_Val.SumError = 0x1200;//启动电 机系数
                        SetOK_Flag = 1;//设定值
                        sys.Run_Status = 1;
                        sys.SetMode_Option = 0;
                        Speed_ADDMode = 0;
                    }
                    else
                    {
                        sys.Motor_Stop = 1;//检测电机
                        Speed_ADDMode = 2;//进入减速模式下
                    }
                }
                else
                {
                    Beep_Flash = 7;
                    sys.Lock_On = 1;
                }
                Beep_Time = 0.1;//蜂鸣器响 0.1S
            }
            Key_Flag4 = 0;//按键事件结束，等待下一次按下
            LongPress4 = 0;//长按标志清零
            Key_Cnt4 = 0;//按钮计数清零
        }
        if(Key_Cnt4 > 1.5 && Key_Cnt4 < 3)//按键时间大于 1.5S 小于 3S 表示长按
        {
            if(LongPress4 == 0)//如果没有一直一直长按着
            {
                Beep_Time = 0.1;//蜂鸣器响 0.1S
                LongPress4 = 1;//长按标志置一
            }
```

```
        }
    }
    ROW1_H;
    ROW2_H;
    ROW3_L;
    /************************************ 加 键
************************************/
    if(HAL_GPIO_ReadPin(COL1_GPIO_Port,COL1_Pin) == 0)//按下按键
    {
        if(sys.Run_Status == 1)
            return;
        Key_Cnt5 += dT;//按下时间++
        Key_Flag5 = 1;//按键按下标志置一
    }
    if(Key_Flag5 == 1)//按键被按下
    {
        if(HAL_GPIO_ReadPin(COL1_GPIO_Port,COL1_Pin) == 1)//抬起按键
        {
            if(Key_Cnt5 < 1.5)//小于 1.5S 是单击
            {
                if(sys.SetMode_Option == 2)//设置时间
                {
                    if(Time_Unit == 0)
                        Set_Time += 10;//时间加 10S
                    else
                        Set_Time += 60;//时间加 60S
                    if(Set_Time > 3590)//设置时间大于 59 分 50 秒时
                        Set_Time = 3590;//设置时间等于 59 分 50 秒时
                }
                else if(sys.SetMode_Option == 3)//设置速度
                {
                    if(Speed_Unit)
                    {
                        Set_Speed += 100;//离心率加 100
                        if(Set_Speed > Xg_MAX)//离心率大于 2100 时
                            Set_Speed = Xg_MAX;//离心率等于 2100
                    }
                    else
                    {
                        Set_Speed += 100;//速度加 100
                        if(Set_Speed > Speed_MAX)//速度大于 2500 时
                            Set_Speed = Speed_MAX;//速度等于 2500
                    }
                }
                else if(sys.SetMode_Option == 4)//设置温度
                {
                    Safe_Set_Temp += 10;//温度加 1 度
                    if(Safe_Set_Temp > 500)//设定安全温度大于 50 度时
                        Safe_Set_Temp = 500;//设定安全温度等于 50 度
                }
```

```
            else if(sys.SetMode_Option == 5)//设置离心率模块
            {
                Speed_Mode += 1;//模块类型加一
                if(Speed_Mode > 6)//模块类型大于六
                    Speed_Mode = 6;//模块类型等于六
            }
            if(Show_Circle == 1 && sys.SetMode_Option == 1)
            {
                PMode_Option++;
                if(PMode_Option > 9)
                {
                    PMode_Option = 9;
                }
                Param_Read();
            }
            Key_Status = 2;//设置时 2S 不闪烁
            Beep_Time = 0.1;//蜂鸣器响 0.1S
            Twinkle_Time = 6;//闪烁时间 6S
        }
        Key_Flag5 = 0;//按键事件结束，等待下一次按下
        Key_Cnt5 = 0;//按钮计数清零
    }
    if(Key_Cnt5 > 1.9 && Key_Cnt5 < 2.1)//按键时间大于 1.9S 小于 2.1S 表示长按
    {
        if(sys.SetMode_Option == 2)//设置时间
        {
            if(Time_Unit == 0)
                Set_Time += 60;//时间加 10S
            else
                Set_Time += 600;//时间加 60S
            if(Set_Time > 3590)//设置时间大于 59 分 50 秒时
                Set_Time = 3590;//设置时间等于 59 分 50 秒时
        }
        else if(sys.SetMode_Option == 3)//设置速度
        {
            if(Speed_Unit)
            {
                Set_Speed += 1000;//离心率加 1000
                if(Set_Speed > Xg_MAX)//离心率大于 2100 时
                    Set_Speed = Xg_MAX;//离心率等于 2100
            }
            else
            {
                Set_Speed += 1000;//速度加 1000
                if(Set_Speed > Speed_MAX)//速度大于 2500 时
                    Set_Speed = Speed_MAX;//速度等于 2500
            }
        }
        else if(sys.SetMode_Option == 4)//设置温度
        {
```

```
                Safe_Set_Temp += 100;//温度加 1 度
                if(Safe_Set_Temp > 500)//设定安全温度大于 50 度时
                    Safe_Set_Temp = 500;//设定安全温度等于 50 度
            }
            else if(sys.SetMode_Option == 5)//设置离心率模块
            {
                Speed_Mode += 1;//模块类型加一
                if(Speed_Mode > 6)//模块类型大于六
                    Speed_Mode = 6;//模块类型等于六
            }
            Key_Status = 2;//设置时 2S 不闪烁
            Key_Flag5 = 0;//按键事件结束，等待下一次按下
            Key_Cnt5 = 1.5;//按钮计数清零
            Beep_Time = 0.1;//蜂鸣器响 0.1S
            Twinkle_Time = 6;//闪烁时间 6S
        }
    }
    /*************************************OPEN                                    键
*************************************/
    if(HAL_GPIO_ReadPin(COL2_GPIO_Port,COL2_Pin )== 0)//按下按键
    {
        if(sys.Run_Status)//如果在显示离心力的模式下都不能操作
            return;
        Key_Cnt6 += dT;//按下时间++
        Key_Flag6 = 1;//按键按下标志置一
    }
    if(Key_Flag6 == 1)//按键被按下
    {
        if(HAL_GPIO_ReadPin(COL2_GPIO_Port,COL2_Pin) == 1)//抬起按键
        {
            if(Key_Cnt6 < 1.5)//小于 1.5S 是单击
            {
                if(HAL_GPIO_ReadPin(UC_IN1_GPIO_Port,UC_IN1_Pin)==  1)//电磁锁  1
闭合时
                    Lock1_Status = 1;//打开电磁锁 1
                if(HAL_GPIO_ReadPin(UC_IN2_GPIO_Port,UC_IN2_Pin)==  1)//电磁锁  2
闭合时
                    Lock2_Status = 1;//打开电磁锁 2
                sys.Run_Status = 0;
                Beep_Time = 0.1;//蜂鸣器响 0.1S
            }
            Key_Flag6 = 0;//按键事件结束，等待下一次按下
            LongPress6 = 0;//长按标志清零
            Key_Cnt6 = 0;//按钮计数清零
        }
        if(Key_Cnt6 > 1.5 && Key_Cnt6 < 3)//按键时间大于 1.5S 小于 3S 表示长按
        {
            if(LongPress6 == 0)//如果没有一直一直长按着
            {
                LongPress6 = 1;//长按标志置一
```

```
                }
            }
        }
}
#include "Drv_Lock.h"

/**********全局变量声明******/
uint8_t Lock1_Status,Lock2_Status;//电磁铁的状态

/*
**********************************************************************
 * 函数原型：void Ctrl_Lock(float dT)
 * 功      能：电磁铁控制
**********************************************************************
*/
void Ctrl_Lock(float dT)
{
    if(Lock1_Status == 1)
    {
        Lock1_ON;//打开电磁锁 1
        Lock1_Status = 0;
    }
    else
    {
        Lock1_OFF;//关闭电磁锁 1
    }
    if(Lock2_Status == 1)
    {
        Lock2_ON;//打开电磁锁 2
        Lock2_Status = 0;
    }
    else
    {
        Lock2_OFF;//关闭电磁锁 2
    }
    if((HAL_GPIO_ReadPin(UC_IN1_GPIO_Port,UC_IN1_Pin)==
1)&&(HAL_GPIO_ReadPin(UC_IN2_GPIO_Port,UC_IN2_Pin)==  1))//电磁锁 1 和电磁锁 2 都
闭合时
    {
        Lid_State = 0;//关闭盖子，显示图标
    }
    else
    {
        Lid_State = 1;//打开盖子，显示图标
    }

}
#include "Drv_HT1623.h"

/*
```

```
*********************************************************************
 * 函数原型：static void delay(uint16_t time)
 * 功      能：us 延时
 * 输      入: time ：时间
 * 参      数：uint16_t time
 * 调      用：内部调用
*********************************************************************
*/
static void delay(uint16_t time)
{
    unsigned char a;
    for(a = 100; a > 0; a--);
}

/*
*********************************************************************
 * 函数原型：static void Write_Mode(unsigned char MODE)
 * 功      能：写入模式,数据 or 命令
 * 输      入: MODE ：数据 or 命令
 * 参      数：unsigned char MODE
 * 调      用：内部调用
*********************************************************************
*/
static void Write_Mode(unsigned char MODE)
{
    delay(10);
    Clr_1625_Wr;//RW = 0;
    delay(10);
    Set_1625_Dat;//DA = 1;
    Set_1625_Wr;//RW = 1;
    delay(10);
    Clr_1625_Wr;//RW = 0;
    delay(10);
    Clr_1625_Dat;
    delay(10);//DA = 0;
    Set_1625_Wr;//RW = 1;
    delay(10);
    Clr_1625_Wr;//RW = 0;
    delay(10);
    if (0 == MODE)
    {
        Clr_1625_Dat;//DA = 0;
    }
    else
    {
        Set_1625_Dat;//DA = 1;
    }
    delay(10);
    Set_1625_Wr;//RW = 1;
    delay(10);
```

```
}

/*
*********************************************************************
 * 函数原型：static void Write_Command(unsigned char Cbyte)
 * 功      能：LCD 命令写入函数
 * 输      入: Cbyte：控制命令字
 * 参      数：unsigned char Cbyte
 * 调      用：内部调用
*********************************************************************
*/
static void Write_Command(unsigned char Cbyte)
{
    unsigned char i = 0;
    for (i = 0; i < 8; i++)
    {
        Clr_1625_Wr;
        //Delay_us(10);
        if ((Cbyte >> (7 - i)) & 0x01)
        {
            Set_1625_Dat;
        }
        else
        {
            Clr_1625_Dat;
        }
        delay(10);
        Set_1625_Wr;
        delay(10);
    }
    Clr_1625_Wr;
    delay(10);
    Clr_1625_Dat;
    Set_1625_Wr;
    delay(10);
}

/*
*********************************************************************
 * 函数原型：static void Write_Address(unsigned char Abyte)
 * 功      能：LCD 地址写入函数
 * 输      入: Abyte：地址
 * 参      数：unsigned char Abyte
 * 调      用：内部调用
*********************************************************************
*/
static void Write_Address(unsigned char Abyte)
{
    unsigned char i = 0;
    Abyte = Abyte << 1;
```

```
    for (i = 0; i < 6; i++)
    {
        Clr_1625_Wr;
        if ((Abyte >> (6 - i)) & 0x01)
        {
            Set_1625_Dat;
        }
        else
        {
            Clr_1625_Dat;
        }
        delay(10);
        Set_1625_Wr;
        delay(10);
    }
}

/*
*********************************************************************
 * 函数原型：static void Write_Data_8bit(unsigned char Dbyte)
 * 功      能：LCD 8bit 数据写入函数
 * 输      入: Dbyte：数据
 * 参      数：unsigned char Dbyte
 * 调      用：内部调用
*********************************************************************
*/
static void Write_Data_8bit(unsigned char Dbyte)
{
    int i = 0;
    for (i = 0; i < 8; i++)
    {
        Clr_1625_Wr;
        if ((Dbyte >> (7 - i)) & 0x01)
        {
            Set_1625_Dat;
        }
        else
        {
            Clr_1625_Dat;
        }
        delay(10);
        Set_1625_Wr;
        delay(10);
    }
}

/*
*********************************************************************
 * 函数原型：void Write_Data_4bit(unsigned char Dbyte)
 * 功      能：LCD 4bit 数据写入函数
```

```
 * 输      入: Dbyte：数据
 * 参      数：unsigned char Dbyte
 * 调      用：内部调用
**********************************************************************
*/
void Write_Data_4bit(unsigned char Dbyte)
{
      int i = 0;
      for (i = 0; i < 4; i++)
      {
            Clr_1625_Wr;
            if ((Dbyte >> (3 - i)) & 0x01)
            {
                  Set_1625_Dat;
            }
            else
            {
                  Clr_1625_Dat;
            }
            delay(10);
            Set_1625_Wr;
            delay(10);
      }
}

/*
**********************************************************************
 * 函数原型：void Lcd_Init(void)
 * 功      能：LCD 初始化，对 lcd 自身做初始化设置
**********************************************************************
*/
void Lcd_Init(void)
{
      Set_1625_Cs;
      Set_1625_Wr;
      Set_1625_Dat;
      delay(500);
      Clr_1625_Cs;//CS = 0;
      delay(10);
      Write_Mode(0);//命令模式
      Write_Command(0x01);//Enable System
      Write_Command(0x03);//Enable Bias
      Write_Command(0x04);//Disable Timer
      Write_Command(0x05);//Disable WDT
      Write_Command(0x08);//Tone OFF
      Write_Command(0x18);//on-chip RC 震荡
      Write_Command(0x29);//1/4Duty 1/3Bias
      Write_Command(0x80);//Disable IRQ
      Write_Command(0x40);//Tone Frequency 4kHZ
      Write_Command(0xE3);//Normal Mode
```

```
    Set_1625_Cs;//CS = 1;

    HAL_TIM_PWM_Start(&htim14, TIM_CHANNEL_1);
     __HAL_TIM_SET_COMPARE(&htim14, TIM_CHANNEL_1,60);//不输出 pwm

    Lcd_All();
    HAL_Delay(1000);
    Lcd_Clr();
}

/*
**********************************************************************
 * 函数原型：void Lcd_Clr(void)
 * 功      能：LCD 清屏函数
**********************************************************************
*/
void Lcd_Clr(void)
{
    Write_Addr_Dat_N(0x0, 0x00, 50);
}


/*
**********************************************************************
 * 函数原型：void Lcd_All(void)
 * 功      能：LCD 全显示函数
**********************************************************************
*/
void Lcd_All(void)
{
    Write_Addr_Dat_N(0x0, 0xff, 60);
    Write_Addr_Dat_N(0,0x0f,1);
}

/*
**********************************************************************
 * 函数原型：void Write_Addr_Dat_N(unsigned char _addr, unsigned char _dat, unsigned char
n)
 * 功      能：屏幕显示
 * 输      入: _addr：地址   char _dat：数据  n：个数
 * 参      数：unsigned char _addr, unsigned char _dat, unsigned char n
**********************************************************************
*/
void Write_Addr_Dat_N(unsigned char _addr, unsigned char _dat, unsigned char n)
{
    unsigned char i = 0;
    Clr_1625_Cs;//CS = 0;
    delay(10);
    Write_Mode(1);
    Write_Address(_addr);
```

```
    for (i = 0; i < n; i++)
    {
        Write_Data_8bit(_dat);
    }
    Set_1625_Cs;//CS = 1;
}
#include "Drv_NTC.h"

/**********局部变量**********/
const uint16_t R100K_TAB[] = //R25=100K B25=3950K -25-150
{
387,
407,
429,
451,
474,
498,
523,
549,
575,
603,
631,
660,
691,
722,
754,
787,
820,
855,
890,
927,
964,
1002,
1040,
1079,
1119,
1160,
1201,
1244,
1286,
1329,
1373,
1417,
1461,
1506,
1551,
1596,
1641,
1686,
1732,
```

1777,
1823,
1868,
1913,
1958,
2003,
2048,
2093,
2137,
2181,
2225,
2269,
2312,
2354,
2396,
2438,
2479,
2519,
2559,
2598,
2636,
2674,
2711,
2748,
2784,
2819,
2853,
2887,
2920,
2952,
2984,
3014,
3044,
3074,
3102,
3130,
3157,
3184,
3209,
3235,
3259,
3283,
3306,
3329,
3351,
3372,
3393,
3413,
3433,
3452,

```
3470,
3488,
3506,
3523,
3540,
3556,
3571,
3587,
3601,
3616,
3629,
3643,
3656,
3668,
3681,
3693,
3704,
3715,
3726,
3736,
3746,
3756,
3766,
3775,
3784,
3793,
3801,
3809,
3817,//97
3825,//98
3832,//99
3839,//100
3846,//101
3853,//102
3860,//103
3866,//104
3872,//105
3878,//106
3884,//107
3890,//108
3895,//109
3901,//110
3906,//111
3911,//112
3916,//113
3920,//114
3925,//115
3929,//116
3934,//117
3938,//118
```

```
3942,//119
3946,//120
};

/**********全局变量**********/
uint16_t ADC_Val;//adc 的值
uint8_t NTC_Res;//温度采样返回的状态

/*
*********************************************************************
 * 函数原型：  int Filter_ADC(void)
 * 功      能：  滑动平均值滤波
 * 输      出：  滤波后的值
*********************************************************************
*/
#define N 100//采集 100 次
int ADCvalue_Buf[N];//用于储存采集到的 adc 值
int i = 0;
int Filter_ADC(void)
{
    char count;
    long sum = 0;
    HAL_ADC_Start(&hadc);//开始读取 adc 的值
    ADCvalue_Buf[i++] = HAL_ADC_GetValue(&hadc);//将 adc 的值储存
    if (i == N)//加入读了 100 组就从新开始
    {
        i = 0;
    }
    for (count = 0; count < N; count++)
    {
        sum += ADCvalue_Buf[count];//100 组相加
    }
    if(ADCvalue_Buf[99] == 0)//如果没有读到 100 组就用第一次读到的数
        return ADCvalue_Buf[0];
    else//读到 100 组后
        return (int)(sum / N);//输出平均值
}

/*
*********************************************************************
 * 函数原型：  uint16_t Get_Ntc_Temp(uint16_t value_adc)
 * 功      能：  计算出 Ntc 的温度
 * 输      入:  value_adc:adc 读到的值
 * 参      数：  uint16_t value_adc
*********************************************************************
*/
#define SHORT_CIRCUIT_THRESHOLD 15
#define OPEN_CIRCUIT_THRESHOLD 4096
uint8_t index_l, index_r ;
uint16_t Get_Ntc_Temp(uint16_t value_adc)
```

```
{
    uint8_t R100k_Tab_Size = 141;
    int temp = 0;
    if(value_adc <= SHORT_CIRCUIT_THRESHOLD)
    {
        return 1;
    }
    else if(value_adc >= OPEN_CIRCUIT_THRESHOLD)
    {
        return 2;
    }
    else if(value_adc < R100K_TAB[0])
    {
        return 3;
    }

    else if(value_adc > R100K_TAB[R100k_Tab_Size - 1])
    {
        return 4;
    }

    index_l = 0;
    index_r = R100k_Tab_Size - 1;
    for(; index_r - index_l > 1;)
    {
        if((value_adc <= R100K_TAB[index_r]) && (value_adc > R100K_TAB[(index_l + index_r) % 2 == 0 ? (index_l + index_r) / 2 : (index_l + index_r) / 2 ]))
        {
            index_l = (index_l + index_r) % 2 == 0 ? (index_l + index_r) / 2 : (index_l + index_r) / 2 ;
        }
        else
        {
            index_r = (index_l + index_r) / 2;
        }
    }
    if(R100K_TAB[index_l] == value_adc)
    {
        temp = (((int)index_l) - 20) * 10; //rate *10
    }
    else if(R100K_TAB[index_r] == value_adc)
    {
        temp = (((int)index_r) - 20) * 10; //rate *10
    }
    else
    {
        if(R100K_TAB[index_r] - R100K_TAB[index_l] == 0)
        {
            temp = (((int)index_l) - 20) * 10; //rate *10
        }
```

```
            else
            {
                temp = (((int)index_l) - 20) * 10 + ((value_adc - R100K_TAB[index_l] ) * 100 + 5)
/ 10 / (R100K_TAB[index_r] - R100K_TAB[index_l]);
            }
        }

        /******************温度补偿**********************/
        Safe_Rel_Temp = temp;//测得温度赋值
        return 0;
}

/*
*********************************************************************
 * 函数原型：  void Read_Temp(float dT)
 * 功      能：  读取温度-10ms
*********************************************************************
*/
void Read_Temp(float dT)
{
        static float T;
        T += dT;
        ADC_Val = Filter_ADC();//滤波获取 adc 的滑动平均值
        if(T >= 1)//1S
        {
            NTC_Res = Get_Ntc_Temp(ADC_Val);//计算温度
            T = 0;
        }
}

#include "PID.h"

/**********结构体*************/
PID_val_t Speed_Val;//pid 数据结构
PID_arg_t Speed_Arg;//pid 数据系数

/*
*********************************************************************
 * 函数原型：  void PID_Init(void)
 * 功      能：  pid 系数初始化
*********************************************************************
*/
void PID_Init(void)
{
        Speed_Arg.Kp=0.014;
        Speed_Arg.Ki=0.0035;
        Speed_Arg.Kd=0.005;
}

/*
```

```
**********************************************************************
 * 函数原型：  void PID_Speed(
                uint16_t Expect,    //期望值（设定值）
                uint16_t Feedback, //反馈值（实际值）
                PID_arg_t *pid_arg,//PID 参数结构体
                PID_val_t *pid_val)//PID 数据结构体
 * 功      能：  PID 控制
 * 输      入:  Expect,    //期望值（设定值）
                 Feedback, //反馈值（实际值）
                 PID_arg_t *pid_arg,//PID 参数结构体
                 PID_arg_t *pid_arg,//PID 参数结构体
**********************************************************************
*/
void PID_Speed(
                uint16_t Expect,    //期望值（设定值）
                uint16_t Feedback, //反馈值（实际值）
                PID_arg_t *pid_arg,//PID 参数结构体
                PID_val_t *pid_val)//PID 数据结构体

{
    pid_val->Error = Expect - Feedback;//当前误差
    if(sys.Motor_Stop == 0)
    {
        if(pid_val->Error > 200)
        {
            pid_val->Error = 200;
        }
    }
    else
    {
        if(pid_val->Error < -200)
        {
            pid_val->Error = -200;
        }
    }
    pid_val->SumError = pid_val->Error + pid_val->SumError;//误差和
    pid_val->D_Error = pid_val->Error - pid_val->LastError;//误差偏差
    pid_val->LastError = pid_val->Error;//保存上一次误差
    pid_val->Out                                                    =
pid_arg->Kp*pid_val->Error+pid_arg->Ki*pid_val->SumError+pid_arg->Kd*pid_val->D_Error;
    if(pid_val->Out < 7)
        pid_val->Out = 7;
    if(pid_val->Out > 7 && pid_val->Out<398)
        pid_val->Out=pid_val->Out;
    if(pid_val->Out > 398)
        pid_val->Out = 398;
}
```