

MC4100\_10pro 软件源程序

---

```

#include "main.h"
#include "adc.h"
#include "tim.h"
#include "gpio.h"
#include "ht1623.h"
#include "lcd.h"
#include "user.h"
#include "key.h"
extern uint16_t Rpm,Time_SUM,RUN_Status;
extern uint8_t Set_Flag1,Set_Flag2,Set_Flag3,Set_Flag4,rpm_flag;
uint8_t time_blink=0,set_time_flag;
extern uint16_t rel_rpm;
extern uint8_t run_mode;
extern uint8_t mode_blink;
extern uint8_t p_dis_time;
uint16_t OPEN_Count=0;
void PWM_RPM_Convert(void);
void P_SET(void);
void MENU_SET(void);
void UP_KEY(void);
void DOWN_KEY(void);
void START_KEY(void);
void OPEN(void);
void save_data(void);
void p_circle(void);
#define speedx100 1
void SystemClock_Config(void);
extern uint8_t cnt;
extern uint16_t BEEP_Count,BEEP_Close;
/* USER CODE END PFP */
uint32_t WriteFlashData = 0x12345678;
uint32_t addr = 0x0807E000;
uint16_t Rpm_Cnt,PWM;
extern uint16_t Set_Flag,Set_Count,Key_Count,Key1_Count;
extern uint8_t Cover_Status;
uint16_t Convert_Set;
int set_temp;
extern uint8_t P_Mode_Status;
extern uint8_t circle_run;
extern uint8_t L1_S,L2_S,L3_S,L4_S,L5_S,L6_S,L7_S,L8_S,L9_S,L10_S,L11_S,L12_S;
uint8_t circle_status;
uint8_t circel_count;
int main(void)
{

    HAL_Init();
    SystemClock_Config();
    MX_GPIO_Init();
    MX_TIM1_Init();
    MX_ADC_Init();

```

---

```

    HAL_TIM_Base_Start_IT(&htim1);

Sys_Init();
    lcd_clr();
    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
    HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_3);
    // PWM=0;
    rpm_flag=1;
    __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_3,50);
while (1)
{
    //HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, GPIO_PIN_SET);
    Cover_Status=HAL_GPIO_ReadPin (UP_KEY_GPIO_Port,UP_KEY_Pin);
    if((RUN_Status ==Sys_RUN)||(RUN_Status ==Sys_Down))
    {
        PWM=Rpm/100;

        if(Cover_Status==0)
        {
            PWM=0;
            rel_rpm=0;
            if(RUN_Status ==Sys_RUN)
            {
                BEEP();
                BEEP_Close=200;
            }

            Convert_Set=0;
            RUN_Status =Sys_STOP;
        }
        uint16_t Set_PWM;
        if(Convert_Set>60)
            Set_PWM=Convert_Set;
        else
            Set_PWM=Convert_Set;
        //PWM_RPM_Convert();

        __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_1,Set_PWM+40);//pwm 0—71
speed 0-7000rpm
    }
    else
        __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_1,0);//pwm 0—400
    LCD_Display();
    Key_Handle();
    p_circle();
}
}
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};

```

---

```

RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

/** Initializes the CPU, AHB and APB busses clocks
 */
RCC_OscInitStruct.OscillatorType                                =
RCC_OSCILLATORTYPE_HSI14|RCC_OSCILLATORTYPE_HSE;
RCC_OscInitStruct.HSEState = RCC_HSE_ON;
RCC_OscInitStruct.HSI14State = RCC_HSI14_ON;
RCC_OscInitStruct.HSI14CalibrationValue = 16;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL4;
RCC_OscInitStruct.PLL.PREDIV = RCC_PREDIV_DIV1;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
    Error_Handler();
}
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                               |RCC_CLOCKTYPE_PCLK1;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
{
    Error_Handler();
}
}

/* USER CODE BEGIN 4 */
uint32_t next,Speed_Rel;
uint16_t KEY_Wait;
extern                                     uint16_t
Rpm,Time_SUM,Dis_rpm,Dis_Set_Time,Dis_Rel_Time,Dis_rpm_rel,Dis_rpm_set;
extern uint8_t KEY_Status;
double pwm_data;
uint16_t menu_count,up_count,down_count,p_count;
uint8_t rpm_dis_mode=0;
uint8_t run_mode_set;//跑梯度设置位
uint8_t p_run_mode=0;//跑梯度模式
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    //矩阵键盘
    if(GPIO_Pin ==COL1_Pin)
    {
        if(KEY_Wait==0)
        {
            if(KEY_Status==1)
            {

```

---

```

        P_SET();
        p_count++;
        if(p_count>5)
        {
            p_count=0;
            if(p_run_mode)
            {
                p_run_mode=0;
                run_mode_set=0;
            }
            else
            {
                p_run_mode=1;
            }
        }
        KEY_Wait=400;
    }
    else if(KEY_Status==2)
    {
        MENU_SET();
        menu_count++;
        if(menu_count>5)
        {
            menu_count=0;
            if(rpm_dis_mode==0)
                rpm_dis_mode=1;
            else
                rpm_dis_mode=0;
        }
        KEY_Wait=400;
    }
    else if(KEY_Status==3)
    {
        UP_KEY();
        up_count++;
        if(up_count>5)
        {
            KEY_Wait=50;
        }
        else
            KEY_Wait=400;
    }
}

}

}
else if(GPIO_Pin ==COL2_Pin)
{

```

---

```

    if(KEY_Wait)
    return;
    if(KEY_Wait==0)
    {
        if(KEY_Status==1)
        {
            if(RUN_Status ==Sys_RUN)
            {
                RUN_Status =Sys_Down;

                rpm_flag=1;
                BEEP();
            }
            else
                OPEN();
            //          circle_start=0;
            //OPEN();
            KEY_Wait=400;
        }
        else if(KEY_Status==2)
        {
            START_KEY();
            KEY_Wait=400;
        }
        else if(KEY_Status==3)
        {
            DOWN_KEY();
            down_count++;
            if(down_count>5)
            {
                KEY_Wait=50;
            }
            else
                KEY_Wait=400;
        }
        //KEY_Wait=400;
    }
}

void read_data(void);
uint32_t ms10;
uint32_t ms;
uint16_t val;
uint8_t res;
extern uint16_t Rpm,Time_SUM,Set_Time;
extern uint16_t save_time,save_rpm;
extern uint8_t Point_Flag;
extern uint8_t Sys_Mode;
extern uint16_t BEEP_Count,BEEP_Close;
uint16_t Half_Sec;

```

---

```

extern uint16_t full_rpm;
extern uint16_t full_Convert_Set;
int value_temp;
uint16_t MAX_RPM;
extern uint8_t Time_Status;//0:miao 1:fenzhong
uint8_t KEY1_Pin_ON;
uint16_t UCT_FLAG;
uint8_t circle_start;//中间旋转图标
extern int rel_temp;
extern uint8_t safe_status;
uint8_t safe_flag;
uint8_t safe_beep_status=0;
uint8_t safe_beep_count=0;
uint8_t run_mode1_blink,run_mode2_blink;
extern uint8_t run_mode,run_mode1,run_mode2;
int temp_count;
//extern int rel_temp;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if(htim->Instance == TIM1)
    {
        ms10++;
        if(Key1_Count)
            Key1_Count--;
        if(KEY_Wait)
            KEY_Wait--;

        if(rel_temp>set_temp)
        {
            safe_status=1;
            RUN_Status =Sys_Down;
            rpm_flag=1;
            circle_start=0;
            if(safe_beep_status==0)
            {
                safe_beep_status=1;
                safe_beep_count=10;
            }
        }
        else
        {
            if(set_temp-rel_temp>10)
                safe_status=0;
            safe_beep_status=0;
        }
    }
}

```

---

```

if(UCT_FLAG)
    UCT_FLAG--;
if(UCT_FLAG==0)
    HAL_GPIO_WritePin(UCT_GPIO_Port, UCT_Pin, GPIO_PIN_RESET);

if(BEEP_Close)
    BEEP_Close--;

if(BEEP_Count)
    BEEP_Count--;
if(BEEP_Count==0)
    HAL_GPIO_WritePin(BEEP_GPIO_Port, BEEP_Pin, GPIO_PIN_RESET);
ms++;

if((ms%50)==0)
{
    if( RUN_Status ==7)
    {

        if(rel_rpm>0)
        {

            rel_rpm=rel_rpm-100;
            if(rel_rpm<500)
                rel_rpm=0;

            if(rel_rpm)
                get_pwm(rel_rpm);
            Convert_Set=full_Convert_Set;
            if(rel_rpm==0)
            {
                RUN_Status =Sys_STOP;
                //OPEN();
                BEEP();
                Convert_Set=0;
                full_Convert_Set=0;
                Rpm=save_rpm;
                Time_SUM=save_time;
            }
        }
    }
}

if(ms>200)
{
    ms=0;
    if(rel_rpm)

```



---

```

    {
        circle_run++;
        if(circle_run>3)
            circle_run=1;

    }
    else
    {
        circle_run=0;
    }

    if( RUN_Status==Sys_RUN)
    {
        if(full_Convert_Set>Convert_Set)
        {
            Convert_Set=Convert_Set+20;
            if(Convert_Set>full_Convert_Set)
                Convert_Set=full_Convert_Set;
            rel_rpm=Rpm*Convert_Set/full_Convert_Set;
        }
        else if(full_Convert_Set<Convert_Set)
        {
            Convert_Set=Convert_Set-20;
            if(Convert_Set<full_Convert_Set)
                Convert_Set=full_Convert_Set;
            rel_rpm=Rpm*Convert_Set/full_Convert_Set;
        }
    }

    if(circle_status==1)
    {
        circel_count++;
        if(circel_count>12)
            circel_count=1;
    }

}

if(ms10>500)
{
    Half_Sec++;

    if((set_time_flag==1)&&(RUN_Status ==Sys_RUN))
    {
        time_blink=~time_blink;
    }
    else
        time_blink=0;
    if(Set_Flag4)
    {

```

```
        if(mode_blink)
            mode_blink=0;
        else
            mode_blink=1;
    }
    else
        mode_blink=0;

    if(safe_status)
    {
        if(safe_flag)
            safe_flag=0;
        else
            safe_flag=1;
    }
    else
    {
        safe_flag=1;
    }

    if(run_mode_set==1)
        run_mode1_blink=~run_mode1_blink;
    else
        run_mode1_blink=0;

    if(run_mode_set==2)
        run_mode2_blink=~run_mode2_blink;
    else
        run_mode2_blink=0;
    if(Half_Sec>1)
    {
        if(safe_beep_count)
        {
            BEEP();
            safe_beep_count--;
        }
        if(RUN_Status ==Sys_RUN)
        {
            if(Time_SUM>0)
            {
                Time_SUM--;
                set_time_flag=1;
            }
        }
        else
            time_blink=0;
        if(Time_SUM==0)
```

```
{
    if(p_run_mode==0)
    {
        if(RUN_Status !=Sys_Down)
            OPEN_Count=20;
        RUN_Status =Sys_Down;
    }
    else
    {
        run_mode=run_mode+1;
        if(run_mode>run_mode2)
        {
            if(RUN_Status !=Sys_Down)
                OPEN_Count=20;
            RUN_Status =Sys_Down;
            run_mode=1;
        }
        else
        {
            read_data();
            get_pwm(Rpm);
        }
    }
}

temp_count++;
//温度获取
if(temp_count>1)
{
    HAL_ADC_Start(&hadc);
    val = HAL_ADC_GetValue(&hadc);
    res = func_get_ntc_temp(val, &value_temp);
    temp_count=0;
}

p_dis_time++;
if(p_dis_time>9)
    p_dis_time=0;

if(OPEN_Count)
{
    OPEN_Count--;
    if(OPEN_Count==1)
        OPEN();
}

Half_Sec=0;
```

```
    }
    //设置位置闪烁
    if(Set_Flag||p_run_mode)
    {
        if(Set_Count)
            Set_Count--;
        else
        {
            Set_Flag1=1;
            Set_Flag2=1;
            Set_Flag3=1;
            Set_Flag4=0;
            Set_Flag=0;
            run_mode_set=0;
            if(p_run_mode==0)
            {
                if(P_Mode_Status)
                    save_data();
            }
        }
    }
    if(Set_Flag==1)//时间
    {
        if(Set_Flag1)
            Set_Flag1=0;
        else
            Set_Flag1=1;
    }
    else if(Set_Flag==2)//转速
    {
        if(Set_Flag2)
            Set_Flag2=0;
        else
            Set_Flag2=1;
    }
    else if(Set_Flag==3)//温度
    {
        if(Set_Flag3)
            Set_Flag3=0;
        else
            Set_Flag3=1;
    }
    if(Key_Count)
        Key_Count--;
}

if(RUN_Status ==Sys_RUN)
{
    if(rpm_flag)
        rpm_flag=0;
    else
```

```
        rpm_flag=1;
    }

    ms10=0;

}
}
}
```

```
void P_SET(void)
```

```
{

    if(P_Mode_Status==0)
    {
        read_data();
        P_Mode_Status=1;
        Set_Flag4=1;
        Set_Flag=5;
        Set_Count=20;
    }
    else
    {
        if(Set_Flag==0)
        {
            read_data();
            mode_blink=1;
            Set_Flag=5;
            Set_Flag1=1;
            Set_Flag2=1;
            Set_Flag3=1;
            Set_Flag4=1;
            Set_Count=20;
        }
        else
        {
            save_data();
            Set_Flag=0;
            Set_Flag1=1;
            Set_Flag2=1;
            Set_Flag3=1;
            Set_Flag4=0;
            Set_Count=0;
        }
    }
}
```

```
}
```

```
void MENU_SET(void)
```

```
{
```

---

```

        if(p_run_mode)
        {
            Set_Flag=0;
            run_mode_set++;
            if(run_mode_set>2)
                run_mode_set=1;
        }
        else
        {
            Set_Flag++;
            if(Set_Flag ==1)//时间设定
            {
                Set_Flag1=0;
                Set_Flag2=1;
                Set_Flag3=1;
            }
            else if(Set_Flag ==2)//转速设定
            {
                Set_Flag1=1;
                Set_Flag2=0;
                Set_Flag3=1;
            }
            else if(Set_Flag==3)
            {
                Set_Flag1=1;
                Set_Flag2=1;
                Set_Flag3=0;
            }
            else if(Set_Flag==4)
            {
                Set_Flag=0;
                Set_Flag1=1;
                Set_Flag2=1;
                Set_Flag3=1;
            }
            else if(Set_Flag==6)
            {
                Set_Flag=1;
                Set_Flag1=0;
                Set_Flag2=1;
                Set_Flag3=1;
            }
        }

        Set_Count=10;
        Key1_Count=5000;
    }
    void UP_KEY(void)
    {
        if(p_run_mode)

```

---

```

{
    if(run_mode_set==1)
    {
        run_mode1++;
        if(run_mode1>9)
            run_mode1=1;
        run_mode=run_mode1;
        read_data();
    }
    else if(run_mode_set==2)
    {
        run_mode2++;
        if(run_mode2>9)
            run_mode2=1;
        run_mode=run_mode2;
        read_data();
    }
}
else
{
    if(Set_Flag==1)
    {
        Set_Time +=10;

        if(Set_Time>3590)
            Set_Time=3590;
        Time_SUM=Set_Time;
    }
    else if(Set_Flag==2)
    {
        if(Sys_Mode==Sys_RPM)//rpm 模式
            MAX_RPM=10000;
        else if(Sys_Mode==Sys_RCF)//rcf 模式
            MAX_RPM=10000;
        Rpm=Rpm+100;
        if(Rpm>MAX_RPM)
            Rpm=MAX_RPM;
    }
    else if(Set_Flag==3)
    {
        set_temp=set_temp+10;
        if(set_temp>500)
            set_temp=500;
    }
    else if(Set_Flag==5)
    {
        save_data();
        run_mode++;

        if(run_mode>9)

```

---

```

        run_mode=1;
        read_data();
    }
}

if(RUN_Status==Sys_STOP)
    Set_Count=10;//按键设置计时
else
    Set_Count=3;
    Key_Count=3;//按键加减计时

    get_pwm(Rpm);

}

void DOWN_KEY(void)
{

    if(p_run_mode)
    {
        if(run_mode_set==1)
        {
            run_mode1--;
            if(run_mode1<1)
                run_mode1=9;
            run_mode=run_mode1;
            read_data();
        }
        else if(run_mode_set==2)
        {
            run_mode2--;
            if(run_mode2<1)
                run_mode2=9;
            run_mode=run_mode2;
            read_data();
        }
    }
    else
    {
        if(Set_Flag==1)
        {
            if(Set_Time>10)
                Set_Time -=10;
            Time_SUM=Set_Time;
        }
        else if(Set_Flag==2)
        {
            if(Rpm>1000)
                Rpm=Rpm-100;
        }
    }
}

```



---

```

    }
    else if(Set_Flag==3)
    {
        if(set_temp>rel_temp+50)

            set_temp=set_temp-10;

    }
    else if(Set_Flag==5)
    {
        save_data();
        run_mode--;

        if(run_mode<1)
            run_mode=9;
        read_data();
    }
}

if(RUN_Status==Sys_STOP)
Set_Count=10;//按键设置计时
else
Set_Count=3;
Key_Count=3;//按键加减计时
//PWM=Rpm/30;

get_pwm(Rpm);

}

void START_KEY(void)
{
    if(RUN_Status ==Sys_RUN)
    {
        RUN_Status =Sys_Down;
        rpm_flag=1;
        circle_start=0;
        OPEN_Count=20;
    }
    else
    {
        if(safe_status==0)
        {
            RUN_Status =Sys_RUN;
            circle_start=1;
            if(p_run_mode)
            {
                run_mode=run_mode1;
                read_data();
            }
        }
    }
}

```

```

        save_time=Set_Time;
        save_rpm=Rpm;
        Set_Flag=0;
        Set_Flag3=1;
        Set_Flag2=1;
        Set_Flag1=1;
        Set_Flag4=0;
        run_mode_set=0;
        full_rpm=Rpm;
        get_pwm(Rpm);
        if(p_run_mode==0)
            save_data();
    }
}
BEEP();
}

void OPEN(void)
{

    HAL_GPIO_WritePin(UCT_GPIO_Port, UCT_Pin, GPIO_PIN_SET);
    UCT_FLAG=100;
    BEEP();
}
int save_buf[3];
void flash_write_buf(uint32_t add, int *data, uint8_t len);
#define flash_addr 0x08007000
#define BYTE_SIZE 0x04
void save_data(void)
{
    save_buf [0]=Set_Time;
    save_buf [1]=Rpm;
    save_buf [2]=set_temp;
    flash_write_buf(flash_addr+0x1000*(run_mode-1), save_buf, 3);
}
void read_data(void)
{
    Set_Time=((uint16_t*) (flash_addr+0x1000*(run_mode-1)));
    Rpm=((uint16_t*) (flash_addr+0x1000*(run_mode-1)+4));
    set_temp= ((uint16_t*) (flash_addr+0x1000*(run_mode-1)+8));
    Time_SUM=Set_Time;
    if(Rpm>12500)
    {
        Set_Time=300;
        set_temp=500;
        Rpm=10000;
        Time_SUM=Set_Time;
    }
}

```

```

}
void flash_write(uint32_t add, uint8_t data)
{
    FLASH_EraseInitTypeDef My_Flash;          ///? FLASH_EraseInitTypeDef ???
My_Flash
    HAL_FLASH_Unlock();                        ///?Flash
    My_Flash.TypeErase = FLASH_TYPEERASE_PAGES; ///?Flash????????
    My_Flash.PageAddress = add;                ///????
    My_Flash.NbPages = 1;                      ///????,????Min_Data =
1?Max_Data =(????-????)???

    uint32_t PageError = 0;
    ///?PageError,????????????FLASH?
    HAL_FLASHEx_Erase(&My_Flash, &PageError); ///????

    uint16_t Write_Flash_Data = data;

    /*?Flash???,FLASH_TYPEPROGRAM_HALFWORD ???Flash???16??,???32??64???,???
    HAL????*/
    HAL_FLASH_Program(FLASH_TYPEPROGRAM_HALFWORD, add,
Write_Flash_Data);
    HAL_FLASH_Lock();                          ///?Flash
}
/*
uint8_t flash_read(uint32_t add)
///?:flash????
///?:add-????
///?:temp-????
*/
uint8_t flash_read(uint32_t add)
{
    uint8_t temp;
    temp = *(__IO uint8_t *) (add);
    return temp;
}
void flash_write_buf(uint32_t add, int *data, uint8_t len)
{
    uint8_t i = 0;
    FLASH_EraseInitTypeDef My_Flash;          ///? FLASH_EraseInitTypeDef ???
My_Flash
    HAL_FLASH_Unlock();                        ///?Flash

    My_Flash.TypeErase = FLASH_TYPEERASE_PAGES; ///?Flash????????
    My_Flash.PageAddress = add;                ///????
    My_Flash.NbPages = 1;                      ///????,????Min_Data =
1?Max_Data =(????-????)???

    uint32_t PageError = 0;
    ///?PageError,????????????FLASH?
    HAL_FLASHEx_Erase(&My_Flash, &PageError); ///????

```

---

```

    for (i = 0; i < len; i++)
    {

/*?Flash???,FLASH_TYPEPROGRAM_HALFWORD  ???Flash??16??,???32??64???,???
HAL????*/
        HAL_FLASH_Program(FLASH_TYPEPROGRAM_HALFWORD, (add + i *
BYTE_SIZE), data[i]);
    }

    HAL_FLASH_Lock();                                //??Flash
}

void flash_read_buf(uint32_t add, int *data, uint8_t len)
{
    uint8_t i = 0;
    for (i = 0; i < len; i++)
    {
        data[i] = *(__IO int*)(add + i * BYTE_SIZE);
    }
}

void p_circle(void)
{
    if((p_run_mode)&&((RUN_Status ==Sys_RUN)||((RUN_Status ==Sys_Down)))
    {
        circle_status=1;
        switch (cirlcel_count)
        {
            case 1:L1_S=1;L7_S=0;
                break;
            case 2:L2_S=1;L8_S=0;
                break;
            case 3:L3_S=1;L9_S=0;
                break;
            case 4:L4_S=1;L10_S=0;
                break;
            case 5:L5_S=1;L11_S=0;
                break;
            case 6:L6_S=1;L12_S=0;
                break;
            case 7:L7_S=1;L1_S=0;
                break;
            case 8:L8_S=1;L2_S=0;
                break;
            case 9:L9_S=1;L3_S=0;
                break;
            case 10:L10_S=1;L4_S=0;
                break;

```

---

```
        case 11:L11_S=1;L5_S=0;
            break;
        case 12:L12_S=1;L6_S=0;
            break;
    }

}
else
{
    circl_count=0;
    circle_status=0;
    L1_S=0;
    L2_S=0;
    L3_S=0;
    L4_S=0;
    L5_S=0;
    L6_S=0;
    L7_S=0;
    L8_S=0;
    L9_S=0;
    L10_S=0;
    L11_S=0;
    L12_S=0;
}
}
void Error_Handler(void)
{
}

void assert_failed(char *file, uint32_t line)
{

}
#include "key.h"
#include "user.h"

extern uint16_t Rpm,Time_SUM;
uint16_t Scan_Status,KEY_Flag,RUN_Status;
uint16_t cur,Set_Flag,Set_Count,Key_Count,Key1_Count;
extern uint8_t Set_Flag1,Set_Flag2;
extern uint8_t Time_Status;
extern uint8_t Sys_Mode;
extern uint8_t Point_Flag;
//uint16_t MAX_RPM;

extern uint16_t PWM;
extern uint16_t BEEP_Count,BEEP_Close;
uint16_t full_rpm;
uint16_t full_Convert_Set;
extern uint8_t rpm_flag;
```

```

uint16_t save_time,save_rpm;
uint8_t KEY_Status;
extern uint16_t KEY_Wait;
extern uint16_t menu_count,up_count,down_count,p_count;
void stop(void);
/*****
*
* 名 称: Key_Handle(void)
* 功 能: 按键处理
* 参 数: PIO_TypeDef* GPIOx,uint16_t GPIO_Pin
* 返 回 值:
*
* 修改历史:
* 改动原因:
* -----
*****/
/

void Key_Handle(void)
{
    KEY_Status=1;
    HAL_GPIO_WritePin(ROW3_GPIO_Port,ROW3_Pin,GPIO_PIN_SET);
    HAL_GPIO_WritePin(ROW1_GPIO_Port,ROW1_Pin,GPIO_PIN_RESET);

    HAL_Delay(1);
    if(HAL_GPIO_ReadPin(COL1_GPIO_Port,COL1_Pin)==1)
        p_count=0;
    KEY_Status=2;
    HAL_GPIO_WritePin(ROW1_GPIO_Port,ROW1_Pin,GPIO_PIN_SET);
    HAL_GPIO_WritePin(ROW2_GPIO_Port,ROW2_Pin,GPIO_PIN_RESET);

    HAL_Delay(1);
    if(HAL_GPIO_ReadPin(COL1_GPIO_Port,COL1_Pin)==1)
        menu_count=0;

    KEY_Status=3;
    HAL_GPIO_WritePin(ROW2_GPIO_Port,ROW2_Pin,GPIO_PIN_SET);
    HAL_GPIO_WritePin(ROW3_GPIO_Port,ROW3_Pin,GPIO_PIN_RESET);
    HAL_Delay(1);
    if(HAL_GPIO_ReadPin(COL1_GPIO_Port,COL1_Pin)==1)
    {
        up_count=0;
        //KEY_Wait=400;
    }

    if(HAL_GPIO_ReadPin(COL2_GPIO_Port,COL2_Pin)==1)
    {
        down_count=0;
        //KEY_Wait=400;
    }
}

```

```

}

void stop(void)
{
    RUN_Status = Sys_STOP;

}

#include "user.h"
#include "ht1623.h"
#include "tim.h"
uint8_t Sys_Mode;//系统运行模式
uint8_t Cover_Status;
extern uint8_t P_Mode_Status;//进入模式选择模式
extern uint8_t Time_Status;
extern uint16_t Rpm,Time_SUM,Key_Count;
extern uint16_t cur,KEY_Flag;
extern uint16_t RUN_Status;
uint8_t Point_Flag;
uint16_t BEEP_Count,BEEP_Close;
extern uint8_t run_mode,run_mode1,run_mode2;
int rel_temp;
uint8_t safe_status;
extern uint8_t safe_flag;
extern int set_temp;
extern uint8_t Set_Flag1,Set_Flag2,Set_Flag3;
extern uint16_t Set_Time;
extern void flash_write_buf(uint32_t add, int *data, uint8_t len);
extern void read_data(void);
int save_buf2[3]={ 300,10000,500};
#define flash_addr 0x08007000
//static const uint16_t Rpm_TAB[] = {
//1000,1620,2500,3040,3960,5400,6900,9090,10700,11400,11800,12000,
//};
//static const uint16_t PWM_TAB[] = {
//21,40,60,80,100,150,200,300,400,500,600,700,
//};

static const uint16_t Rpm_TAB[] = {
1000,1400,2200,2700,3400,4000,4600,5900,6940,7700,8350,8900,9400,10200,12000
};
static const uint16_t PWM_TAB[] = {
33,48,80,100,130,160,200,300,400,500,600,700,800,900,1000
};
extern uint16_t full_Convert_Set;
void BEEP(void)
{
    if(BEEP_Close==0)
    {

```

---

```

    HAL_GPIO_WritePin(BEEP_GPIO_Port, BEEP_Pin, GPIO_PIN_SET);

    BEEP_Count=100;
    }

}

void Sys_Init(void)
{
    HAL_GPIO_WritePin(BEEP_GPIO_Port, BEEP_Pin, GPIO_PIN_RESET);
    HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_1);
    Sys_Mode=Sys_RPM;
    RUN_Status=Sys_STOP;
    Point_Flag=0;
    Time_Status =0;
    KEY_Flag=0;
    Key_Count=0;
    run_mode=1;
    run_mode1=1;
    run_mode2=1;
    safe_status=0;
    // Set_Time=((uint16_t*) (flash_addr+0x1000*(run_mode-1)));
    // if(Set_Time>9900)
    // {
    //     flash_write_buf(flash_addr+0x1000*(run_mode-1), save_buf2, 3);
    // }
    // read_data();
    // safe_flag=1;
    //默认数据
    Set_Time=300;
    set_temp=500;
    Rpm=10000;
    Time_SUM=Set_Time;

    cur=400;
    Set_Flag3=1;
    Set_Flag2=1;
    Set_Flag1=1;

    P_Mode_Status=0;
    BEEP_Close=0;
    lcd_all();
    BEEP();
    lcd_init();

}

uint8_t index_l2, index_r2 ;

void get_pwm(uint16_t rpm1)
{

```



---

```

    if(rpm1 < Rpm_TAB[0])
    {
        full_Convert_Set=0;
    }

    else if(rpm1 > Rpm_TAB[14])
    {
        full_Convert_Set=PWM_TAB[14];
    }

    index_l2 = 0;
    index_r2 = 15 - 1;
    for(;index_r2 - index_l2 > 1;)
    {
        if((rpm1 <= Rpm_TAB[index_r2]) && (rpm1 > Rpm_TAB[(index_l2 + index_r2)%2
== 0 ? (index_l2 + index_r2)/2 : (index_l2 + index_r2)/2 ]))
        {
            index_l2 = (index_l2 + index_r2) % 2 == 0 ? (index_l2 + index_r2)/2 : (index_l2 +
index_r2)/2 ;
        }
        else
        {
            index_r2 = (index_l2 + index_r2)/2;
        }
    }
    if(Rpm_TAB[index_l2] == rpm1)
    {
        full_Convert_Set = PWM_TAB[index_l2];
    }
    else if(Rpm_TAB[index_r2] == rpm1)
    {
        full_Convert_Set = PWM_TAB[index_r2];
    }
    else
    {
        if(Rpm_TAB[index_r2] - Rpm_TAB[index_l2] == 0)
        {
            full_Convert_Set = PWM_TAB[index_l2];
        }
        else
        {
            full_Convert_Set = PWM_TAB[index_l2] + (PWM_TAB[index_r2] -
PWM_TAB[index_l2])*(rpm1-Rpm_TAB[index_l2])/(Rpm_TAB[index_r2]
- Rpm_TAB[index_l2]);/(((int)index_l2) - 20)*10 + ((value_adc-R10K_TAB[index_l2] )*100 +
5)/10/(R10K_TAB[index_r2] - R10K_TAB[index_l2]);
        }
    }
}

```