VM4100 源程序

```
#include "main.h"
#include "tim.h"
#include "gpio.h"

#include "ht1623.h"
#include "lcd.h"
#include "user.h"
#include "key.h"
extern uint16_t Time_SUM,RUN_Status;
extern int Rpm,Set_Speed;
void stop(void);
uint32_t P_Status;    //捕获周期计数状态    1 开启  0 关闭
uint8_t CAPTURE_Status=0; //捕获状态
uint16_t    TIM1CH1_CAPTURE_STA=0;   //捕获周期数
uint32_t   TIM1CH1_CAPTURE_VAL;//捕获计数值
uint8_t      CAPTURE_First=0;//捕获第一个高电平
uint16_t Speed1_Flag;//速度调 0 标志位
uint8_t rpm_flag;
extern int save_Rpm;
extern int sumError1;
extern int lastError1;
extern uint16_t       dis_speed_N;
extern uint16_t       dis_speed_F;
extern int save_time;
void SystemClock_Config(void);
extern    uint8_t cnt;
uint32_t WriteFlashData = 0x12345678;
uint32_t addr = 0x0807E000;
uint16_t    Rpm_Cnt,PWM;
uint8_t key_status1,key_status2,key_status3,key_status4;
extern uint16_t Set_Flag,Set_Count,Key_Count,Key1_Count;
extern uint8_t ADD_Mode;//显示增减模式

extern uint16_t dis_rpm;
uint32_t next;
int Speed_Rel,Start_Time;
uint8_t rel_flag;
uint32_t ms10,P_MS,us50;
extern uint16_t Time_SUM;
extern uint8_t Set_Flag1,Set_Flag2;
extern uint8_t Point_Flag;
extern uint8_t Sys_Mode;
extern uint16_t BEEP_Count,BEEP_Close;
extern int Rpm;
uint8_t point_run;
uint16_t Half_Sec;
uint8_t stop_flag;
uint16_t point_wite;
/*FLASH????*/
void writeFlashTest(void)
```

```
{
    /* 1/4??FLASH*/
    HAL_FLASH_Unlock();


    FLASH_EraseInitTypeDef FlashSet;
    FlashSet.TypeErase = FLASH_TYPEERASE_PAGES;
    FlashSet.PageAddress = addr;
    FlashSet.NbPages = 1;


    uint32_t PageError = 0;
    HAL_FLASHEx_Erase(&FlashSet, &PageError);


    HAL_FLASH_Program(FLASH_TYPEPROGRAM_WORD, addr, WriteFlashData);


    HAL_FLASH_Lock();
}
void printFlashTest(void)
{
    uint32_t temp = *(__IO uint32_t*)(addr);

    printf("addr is:0x%x, data is:0x%x\r\n", addr, temp);
}


int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_TIM1_Init();
    HAL_TIM_Base_Start_IT(&htim1);

   Sys_Init();
     __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_2,200);//Set_Speed);//pwm
0—400
     HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
     HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_2);
     HAL_TIM_IC_Start_IT(&htim1, TIM_CHANNEL_3);
      //PWM=20;
  while (1)
  {

        Key_Handle();
        LCD_Display();

        if(CAPTURE_Status)
```

```
{

        __HAL_TIM_ENABLE(&htim1);
        CAPTURE_Status=0;
        TIM1CH1_CAPTURE_STA=0;
}

if((Sys_Mode==Sys_Point)&&(point_wite ==0))
{
                if(HAL_GPIO_ReadPin (GPIOB,KEY_T_Pin) ==0)
                {
                                if(point_run==0)
                                        {
                                                save_Rpm=Rpm;
                Start_Time=1;
                                        sumError1=0x24000;
                                                ADD_Mode =1;
                                                dis_rpm=0;
                                                //rpm_flag =0;
                                        }
                                point_run=1;
                                        RUN_Status =Sys_RUN;


                }
                else if(HAL_GPIO_ReadPin (GPIOB,KEY_T_Pin) ==1)
                {
                        if(RUN_Status ==Sys_RUN)
                        {
                                if(stop_flag==0)
                                stop_flag=5;


                }
                        if(stop_flag ==1)
                        {

                                        point_run=0;
                                        RUN_Status =Sys_STOP;
                                        sumError1=0;
                                        lastError1=0;
                                        Rpm=save_Rpm;
                                        dis_speed_N=0;
                                        dis_speed_F=0;
                                rpm_flag =1;
                        //stop();
                }
        }
}
```

```
//key1
         if(HAL_GPIO_ReadPin (GPIOB,KEY1_Pin) == 0 )
        {
            key_status1=1;
        }
        if(key_status1)
        {
            if(HAL_GPIO_ReadPin (GPIOB,KEY1_Pin) == 1 )
            {

                   BEEP();
           //BEEP_Close=200;


                key_status1=0;
                //at_beep=0;

            }
        }

//key2
         if(HAL_GPIO_ReadPin (GPIOB,KEY2_Pin) == 0 )
        {
            key_status2=1;
        }
        if(key_status2)
        {
            if(HAL_GPIO_ReadPin (GPIOB,KEY2_Pin) == 1 )
            {

                   BEEP();
           //BEEP_Close=200;


                key_status2=0;
                //at_beep=0;

            }
        }
    //key3
         if(HAL_GPIO_ReadPin (GPIOB,KEY3_Pin) == 0 )
        {
            key_status3=1;
        }
        if(key_status3)
        {
            if(HAL_GPIO_ReadPin (GPIOB,KEY3_Pin) == 1 )
            {
```

```
                              BEEP();
                        //BEEP_Close=200;


                            key_status3=0;
                            //at_beep=0;

                        }
                    }

        //key4
                     if(HAL_GPIO_ReadPin (GPIOB,KEY4_Pin) == 0 )
                    {
                        key_status4=1;
                    }
                    if(key_status4)
                    {
                        if(HAL_GPIO_ReadPin (GPIOB,KEY4_Pin) == 1 )
                        {

                              BEEP();
                        //BEEP_Close=200;


                            key_status4=0;
                            //at_beep=0;

                        }
                    }



    }

}

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_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  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();
   }
   /** Initializes the CPU, AHB and APB busses clocks
   */
   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();
   }
}
extern uint8_t time_free_mode;
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
   if(htim->Instance == TIM1)
   {
            //50us

            ms10++;
            us50++;
            P_MS++;


            if(RUN_Status ==Sys_DOWN )
            {
                  if(Speed_Rel<100)
                  {
                        stop();
                  }
      }



            if(P_Status)
            {

                  TIM1CH1_CAPTURE_STA++;
              }




            if(Key1_Count)
            Key1_Count--;
```

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

      if(P_MS>300)//400
      {





if(RUN_Status ==Sys_RUN)
            {
                if(Start_Time)
              {
                      __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_1,50);
              }
              else
              PWM_Set();


              if(stop_flag)
                      stop_flag --;



        }
            else if(RUN_Status==Sys_STOP)
                      __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_1,0);



            P_MS=0;
    }
      if(us50>5000)
      {
            if(Start_Time)
            Start_Time--;
            us50=0;
      }

    if(ms10>10000)//500ms
    {

            if(Speed1_Flag)
                    Speed1_Flag--;
            if(Speed1_Flag==0)
                    Speed_Rel=0;
                    if(RUN_Status ==Sys_RUN)
                    {
                            if(rpm_flag==0)
                                    rpm_flag=1;
                            else
                                    rpm_flag=0;
                    }

                    Half_Sec++;


                    if(Half_Sec>1)
                            {
                                    if(time_free_mode==0)
                                    {
                                                    if(RUN_Status ==Sys_RUN)
                                                            Time_SUM--;
                                                    if(Time_SUM==0)
                                                    {
                                                            BEEP ();
                                                            RUN_Status=Sys_DOWN;
                                                            //save_Rpm=Rpm;
                                                            Rpm=0;
                                                            ADD_Mode =0;

    __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_1,35);
                                                            Time_SUM=save_time;
                                                            rpm_flag =1;
                                                            if(Sys_Mode==Sys_Point)
                                                            {
                                                                    point_wite=10;
                                                            }
                                                    }
                                        }
                                    Half_Sec=0;
                            }
                            //if(Sys_Mode==Sys_Point)
                    //      Point_Flag=~Point_Flag;
            if(point_wite)
                            point_wite--;

            //设置位置闪烁
if(Set_Flag)
            {
                    if(Set_Count)
                    Set_Count--;
                    else
                    {
                            Set_Flag1=0;
                        Set_Flag2=0;
                        Set_Flag=0;
                    }

                    if(Set_Flag==1)
                    Set_Flag1=~Set_Flag1;
                    else if(Set_Flag==2)
                    Set_Flag2=~Set_Flag2;


                        //rpm_flag=~rpm_flag;

                    if(Key_Count)
                        Key_Count--;
                }
                ms10=0;



            }

        //10ms//0.1ms
    }
}



void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
    if(CAPTURE_Status==0)
    {
         Speed1_Flag=2;
        if(CAPTURE_First)
            {
                CAPTURE_Status=1;        //停止捕获计时
                CAPTURE_First=0;     //清除捕获第一个上升沿标志

    TIM1CH1_CAPTURE_VAL=HAL_TIM_ReadCapturedValue(&htim1,TIM_CHANNEL_3);
//获取当前捕获计数值

                long long temp=0;
                temp=TIM1CH1_CAPTURE_STA;
                temp*=50;    //一个周期 100us
                temp+=TIM1CH1_CAPTURE_VAL; //一个周期所需的 us 数
                temp=30000000/temp; //rpm
                Speed_Rel=temp;
                P_Status=0;
                __HAL_TIM_SET_COUNTER(&htim1,0);
                __HAL_TIM_DISABLE(&htim1);
            }
            else
                  {
                        TIM1CH1_CAPTURE_STA=0;//清除周期计数
                        TIM1CH1_CAPTURE_VAL=0;//清楚捕获寄存器
                        CAPTURE_First=1;                //已捕获第一个上升沿
                        CAPTURE_Status=0;       //捕获计时
                        P_Status=1;       //捕获周期计数
                  }
      }

}


void stop(void)
{
      RUN_Status =Sys_STOP;
      sumError1=0;
      lastError1=0;

            dis_speed_N=0;
            dis_speed_F=0;
      if(Sys_Mode==Sys_Point)//点动模式
      Rpm=save_Rpm;
      else if(Sys_Mode==Sys_Cont)//连续模式
      Rpm=save_Rpm;
      BEEP ();
}
/
void Error_Handler(void)
{

}

void assert_failed(char *file, uint32_t line)
{

}
#endif

#include "user.h"
#include "ht1623.h"
#include "tim.h"
uint8_t Sys_Mode;//系统运行模式
extern uint8_t Time_Status;
extern uint16_t Time_SUM,Key_Count;
extern uint16_t cur,KEY_Flag;
uint8_t Point_Flag;
uint16_t BEEP_Count,BEEP_Close;
extern int Speed_Rel;
int Set_Speed;
```

```
extern int Rpm;
extern uint8_t time_free_mode;
extern uint16_t Scan_Status,KEY_Flag,RUN_Status;
void PID_init(void);
unsigned int PID1(void);
extern uint8_t ADD_Mode;//显示增减模式
extern uint8_t rpm_flag;
void BEEP(void)
{
        if(BEEP_Count==0)
          {
         HAL_GPIO_WritePin(BEEP_GPIO_Port, BEEP_Pin, GPIO_PIN_SET);

       BEEP_Count=3000;
          }

}
extern int save_Rpm;
extern int save_time;
void Sys_Init(void)
{
    HAL_GPIO_WritePin(BEEP_GPIO_Port, BEEP_Pin, GPIO_PIN_RESET);
    HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_1);
   time_free_mode=1;
    //Sys_Mode=Sys_Cont;
    Point_Flag=1;
    Time_Status =0;
    KEY_Flag=0;
    Key_Count=0;
    Time_SUM=300;
    save_time=300;
    Speed_Rel=0;
    Rpm=3000;
    save_Rpm=3000;
    cur=400;
    ADD_Mode=3;
    lcd_all();
    HAL_Delay (1000);
    PID_init();
    BEEP();
    lcd_clr();
    lcd_init();
    Sys_Mode=Sys_Point;
    Point_Flag=0;
    rpm_flag=1;

}
  void PWM_Set(void)
    {
```

```
		if(RUN_Status==Sys_RUN)
		{
//			if(Sys_Mode==Sys_Point)//点动模式
//
	__HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_1,400);//Set_Speed);//pwm
0—400//50-100//
//				else
//				{
					PID1();

	__HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_1,Set_Speed);//Set_Speed);//pw
m 0—400
				//	}
		}
		else if(RUN_Status==Sys_STOP)
				__HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_1,0);//pwm
0—400


	}
struct _pid{

	float Kp,Ki,Kd; //定义比例、积分、微分系数
}pid;
void PID_init(){

	pid.Kp=0.02;//0.6;//3.8
	pid.Ki=0.000388;//0.00088;//0.015
	pid.Kd=0.0001;//0.02
}


int sumError1;
int lastError1;
int B;
unsigned int PID1()
{

//	if(Rpm>1000)
//		pid.Ki=0.000288;
//	else
//		pid.Ki=0.000488;
    int dError=0,Error1=0;
     // if((L1_Rel<2000)&&(L1_Rel>100))
     Error1=Rpm-Speed_Rel;//当前误差
     sumError1=Error1+sumError1;//误差和
         //if(sumError1>3000) sumError1=3000;
     dError=Error1-lastError1;//误差偏差
     lastError1=Error1;
    B=pid.Kp*Error1+pid.Ki*sumError1+pid.Kd*dError;
```

```
                if(B<15)
                        Set_Speed=15;
                        else if(B>200)
                        Set_Speed=200;
             else Set_Speed=B;    //if(B>100&&B<2500)
             Set_Speed=B;
      if(Set_Speed<35) Set_Speed=35;
              return(0);
}
#include "key.h"
#include "user.h"
#include "tim.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;
uint8_t KEY1_Pin_ON;
extern uint16_t PWM;
extern uint16_t BEEP_Count,BEEP_Close;
uint16_t dis_rpm;
extern int Start_Time;
uint8_t ADD_Mode;//显示增减模式
extern int Speed_Rel;
extern uint8_t rpm_flag;
void stop(void);
uint8_t time_free_mode=0;
/******************************************************************************
*
*  名    称: Key_Scan(GPIO_TypeDef* GPIOx,uint16_t GPIO_Pin)
*  功    能:  按键扫描
*  参    数: PIO_TypeDef* GPIOx,uint16_t GPIO_Pin
*  返  回值: KEY_ON/KEY_OFF
*
*  修改历史:
*  改动原因：
*    --------------------------------------------------
*******************************************************************************
/
uint8_t Key_Scan(GPIO_TypeDef* GPIOx,uint16_t GPIO_Pin)
{


             if(HAL_GPIO_ReadPin (GPIOx,GPIO_Pin) == 0 )
             {
                    // BEEP();
                                //BEEP_Close=9000;
                                      if(KEY_Flag==0)
                                      {
                                          KEY_Flag=1;
                                          return KEY_ON;
                                      }
                                 uint32_t cur_time = HAL_GetTick();
                         static uint32_t start_time = 0;
                       if(start_time == 0)
                                       start_time = cur_time;


                                           if(cur_time - start_time < cur)
                                           return KEY_OFF;


                       if(HAL_GPIO_ReadPin (GPIOx,GPIO_Pin) == 0)
                                       {

                                    Scan_Status++;
                                            if(Scan_Status>3)
                                                  cur=18;
                                              start_time = cur_time;
                                              return              KEY_ON;

                                       }

             }
             else
                     {

                          if((HAL_GPIO_ReadPin                         (GPIOB,KEY2_Pin)
==1)&&(HAL_GPIO_ReadPin (GPIOB,KEY3_Pin) ==1 ) )
                          {
                                 if(HAL_GPIO_ReadPin (GPIOB,KEY1_Pin) ==1)
                                   {
                                       KEY1_Pin_ON=0;
                                   }
                                  Scan_Status=0;
                                  cur=400;
                                return           KEY_OFF;
                          }
                     }
                     return           KEY_OFF;
}

/****************************************************************************
*
```

```
* 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：
*    ---------------------------------------------------
*******************************************************************************
/
extern int sumError1;
int save_Rpm;
int save_time;
void Key_Handle(void)
{

                if(( Key_Scan(GPIOB,KEY1_Pin) == KEY_ON))//设置切换按键
                {
                       // BEEP();
              Set_Flag++;

                        Set_Flag1=0;
                        Set_Flag2=0;
                        Set_Count=5;
                        if(Set_Flag>2)
                          Set_Flag=1;
                            KEY1_Pin_ON++;

                            if(KEY1_Pin_ON>3)
                            {
                                Set_Flag=0;
                               HAL_GPIO_WritePin(BEEP_GPIO_Port,     BEEP_Pin,
GPIO_PIN_SET);
                                BEEP_Count=680;
                          BEEP_Close=200;
                                if(Sys_Mode==Sys_Point)
                                {
                                    Sys_Mode=Sys_Cont;
                                    if(Rpm>3000)
                                        Rpm=3000;
                                    Point_Flag=1;
                                }
                                else
                                {
                                    Sys_Mode=Sys_Point;
                                    Point_Flag=0;
                                }
                                KEY1_Pin_ON=0;

                            }
                            Key1_Count=5000;
                            rpm_flag =1;

              }



              if ( (Key_Scan(GPIOB,KEY2_Pin) == KEY_ON))//加键
                {
                      if(Set_Flag==1)
                        {
                            if(Sys_Mode==Sys_Point)//点动模式
                                MAX_RPM=3000;
                            else if(Sys_Mode==Sys_Cont)//连续模式
                                MAX_RPM=3000;
                            Rpm=Rpm+10;
                            if(Rpm>MAX_RPM)
                                Rpm=MAX_RPM;

                                save_Rpm=Rpm;
                            if(Rpm>Speed_Rel)
                            ADD_Mode =1;
                            else
                            ADD_Mode =0;

                        }
                        else if(Set_Flag==2)
                        {
                            if(Time_Status ==0)
                           Time_SUM +=1;
                            else
                             Time_SUM +=60;
                            if(Time_SUM>5940)
                             Time_SUM=5940;

                            time_free_mode=0;
                        }
                            save_time=Time_SUM;
                        Set_Count=5;//按键设置计时
                        Key_Count=3;//按键加减计时

                        //PWM=Rpm/30;
                }


                if ( (Key_Scan(GPIOB,KEY3_Pin) == KEY_ON))//减键
                {


                        if(Set_Flag==1)
```

```
                {
                        Rpm=Rpm-10;
                        if(Rpm<100)
                                Rpm=100;

                                if(Rpm>Speed_Rel)
                        ADD_Mode =1;
                        else
                        ADD_Mode =0;
                                save_Rpm=Rpm;
                }
                else if(Set_Flag==2)
                {
                        if(Time_SUM<61)
                        {
                      Time_SUM -=1;
                                time_free_mode=0;
                                if(Time_SUM<10)
                                {
                                  Time_SUM=10;
                                  time_free_mode=1;
                                }
                        }
                        else
                        {
                                  Time_SUM -=60;
                                time_free_mode=0;
                                if(Time_SUM<10)
                                {
                                  Time_SUM=10;
                                  time_free_mode=1;
                                }
                    }
        }
                Set_Count=5;//按键设置计时
                Key_Count=3;//按键加减计时
                //PWM=Rpm/30;
                save_time=Time_SUM;

}

if ( (Key_Scan(GPIOB,KEY4_Pin) == KEY_ON))
{
        if(RUN_Status!=Sys_DOWN)
        {
        if(Sys_Mode==Sys_Cont)
        {
                if(RUN_Status ==Sys_RUN)
                {
                        RUN_Status=Sys_DOWN;
```

```
                                        //save_Rpm=Rpm;
                                        Rpm=0;
                                        ADD_Mode =0;

    __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_1,35);

                    rpm_flag =1;
                                        //stop();
                                }
                                else
                                {
                                        //if(Sys_Mode==Sys_Cont)
                                        Start_Time=1;
                                        sumError1=0x24000;
                                        RUN_Status =Sys_RUN;
                                        dis_rpm=0;
                                        ADD_Mode =1;
                                        rpm_flag =0;
                                        save_Rpm=Rpm;
                                }



                        }
                }
                }
                        //BEEP();
}
#include "lcd.h"
#include "user.h"
void write_addr_dat_n(unsigned char _addr, unsigned char _dat, unsigned char n);
void speed_deal(void);

uint8_t    LCD_ADD[]={0x5f,0x06,0x3d,0x2f,0x66,0x6b,0x7b,0x0e,0x7f,0x6f};
uint8_t Time_Status;
uint8_t Rpm_B,Rpm_S,Rpm_G,time_1,time_2;
uint16_t Time_SUM,dis;
int Rpm;
extern    uint16_t    Set_Flag,Key_Count,Key1_Count,RUN_Status,dis_rpm;
extern uint8_t Sys_Mode;
uint8_t Set_Flag1,Set_Flag2,point_add;
extern uint8_t Point_Flag;
extern uint8_t KEY1_Pin_ON;
extern int Speed_Rel;
extern uint8_t ADD_Mode;//显示增减模式
uint16_t   dis_speed_N;
uint16_t   dis_speed_F;
extern uint8_t rpm_flag;
extern uint8_t time_free_mode;
void LCD_Display()
```

```
{

    if((RUN_Status ==Sys_RUN)||(RUN_Status ==Sys_DOWN))
    {
        speed_deal();
            dis=Speed_Rel;

            if(ADD_Mode==4)
                dis=Rpm;
            else if(ADD_Mode==3)
                dis=0;
            else
            {
                dis_speed_N=Speed_Rel;
                if(ADD_Mode==1)
                {
                    if(dis_speed_N>dis_speed_F)
                   dis=dis_speed_N;
                    else
                    {
                        dis=dis_speed_F;
                        dis_speed_N=dis_speed_F;
                    }
                }

                if(ADD_Mode==0)
                {
                    if(dis_speed_N<dis_speed_F)
                   dis=dis_speed_N;
                    else
                    {
                        dis=dis_speed_F;
                        dis_speed_N=dis_speed_F;
                    }
                }



                dis_speed_F=dis_speed_N;

            }


    }
        else
      dis=Rpm;
    //Time_SUM=3600;
        if(Set_Flag==1)
            dis=Rpm;
    if(dis<10) {Rpm_B=0;Rpm_S=0;Rpm_G=0;}
    else if (dis<100) {Rpm_B=0;Rpm_S=0;Rpm_G=dis/10;}
    else if (dis<1000) {Rpm_B=0;Rpm_S=dis/100;Rpm_G=dis/10%10;}
    else if (dis<10000) {Rpm_B=dis/1000;Rpm_S=dis/100%10;Rpm_G=dis/10%10;}
    //更新转速

    if(Time_SUM<60) Time_Status =0;
    else Time_Status =1;
    if(Time_Status  ==0)  {time_1=Time_SUM/10;time_2=Time_SUM%10;if(Time_SUM<10)
time_1=0;}
    else                          if(Time_Status                          ==1)
{time_1=Time_SUM/60/10;time_2=Time_SUM/60%10;if(Time_SUM<10) time_1=0;}
    //更新时间



    if(Set_Flag1)
    {
            if((Key_Count==0)&&(Key1_Count==0))
              {
                   write_addr_dat_n(0x00, 0, 1);
                   write_addr_dat_n(0x02, 0, 1);
                   write_addr_dat_n(0x04, 0, 1);
              }
              else
              {
              write_addr_dat_n(0x00, LCD_ADD[Rpm_B], 1);
              write_addr_dat_n(0x02, LCD_ADD[Rpm_S], 1);
              write_addr_dat_n(0x04, LCD_ADD[Rpm_G], 1);
        }

    }
    else
    {
              write_addr_dat_n(0x00, LCD_ADD[Rpm_B], 1);
              write_addr_dat_n(0x02, LCD_ADD[Rpm_S], 1);
              write_addr_dat_n(0x04, LCD_ADD[Rpm_G], 1);
    }

    if(Set_Flag2)
    {
            if((Key_Count==0)&&(Key1_Count==0))
              {
                   if(rpm_flag ==1)
                   {
                        if(time_free_mode==1)
                        {
                             write_addr_dat_n(0x06,0x00|0x80 , 1);
                        }
                        else
                        write_addr_dat_n(0x06,0|0x80, 1);

                }
                else
                {
                        if(time_free_mode==1)
                        {
                                write_addr_dat_n(0x06,0x00&0x7f , 1);
                        }
                        else
                    write_addr_dat_n(0x06,0&0x7f, 1);

                }

                if(Time_Status ==0)
                {
                          if(Point_Flag==1)
                          {
                                          if(time_free_mode==1)
                                                  write_addr_dat_n(0x08, 0x00|0x04, 1);
                                          else
                                                  write_addr_dat_n(0x08,
((((0&0xf)|0x01)&0xf1)&0x7f)|0x04, 1);
                        }
                          else
                          {
                                        if(time_free_mode==1)
                                                write_addr_dat_n(0x08, 0x00|0x02, 1);
                                        else
                                                write_addr_dat_n(0x08, (((0&0xf)|0x01)&0xf1)|0x02, 1);
                        }
                }
                else
                {
                          if(Point_Flag==1)
                          {
                                  if(time_free_mode==1)
                                                  write_addr_dat_n(0x08, 0x00|0x04, 1);
                                  else
                                  write_addr_dat_n(0x08,((((0&0xf)|0x08)&0xf8)&0x7f)|0x04, 1);
                          }
                          else
                          {
                                  if(time_free_mode==1)
                                                write_addr_dat_n(0x08, 0x00|0x02, 1);
                                  else
                                    write_addr_dat_n(0x08, (((0&0xf)|0x08)&0xf8)|0x02, 1);
                          }
                }
                write_addr_dat_n(0x0a, (0&0x0f)<<4, 1);
            }
            else
            {

                if(rpm_flag ==1)
                {
                    if(time_free_mode==1)
                    {
                        write_addr_dat_n(0x06,0x20|0x80 , 1);
                    }
                    else
                  write_addr_dat_n(0x06, LCD_ADD[time_1]|0x80, 1);

                }
                else
                {
                    if(time_free_mode==1)
                    {
                        write_addr_dat_n(0x06,0x20&0x7f , 1);
                    }
                    else
                    write_addr_dat_n(0x06, LCD_ADD[time_1]&0x7f, 1);

                }
                 if(Time_Status ==0)
                     {
                         if(Point_Flag==1)
                         {
                             if(time_free_mode==1)
                                 write_addr_dat_n(0x08, 0x20|0x04, 1);
                             else
                           write_addr_dat_n(0x08,
((((LCD_ADD[time_2]&0xf0)|0x01)&0xf1)&0x7f)|0x04, 1);
                         }
                         else
                         {
                             if(time_free_mode==1)
                                 write_addr_dat_n(0x08, 0x20|0x02, 1);
                             else
                               write_addr_dat_n(0x08,
(((LCD_ADD[time_2]&0xf0)|0x01)&0xf1)|0x02, 1);
                         }
                     }
                     else
                     {
                          if(Point_Flag==1)
                             {
                                 if(time_free_mode==1)
                                   write_addr_dat_n(0x08, 0x20|0x04, 1);
```

```
                                    else
                                        write_addr_dat_n(0x08,
((((LCD_ADD[time_2]&0xf0)|0x08)&0xf8)&0x7f)|0x04, 1);
                                   }
                              else
                                {
                                    if(time_free_mode==1)
                                        write_addr_dat_n(0x08, 0x20|0x02, 1);
                                    else
                                       write_addr_dat_n(0x08,
(((LCD_ADD[time_2]&0xf0)|0x08)&0xf8)|0x02, 1);
                                }
                            }
                            if(time_free_mode==1)
                                   write_addr_dat_n(0x0a, 0x0, 1);
                            else
                              write_addr_dat_n(0x0a, (LCD_ADD[time_2]&0x0f)<<4, 1);
                }

        }
        else
        {
                if(rpm_flag ==1)
                {
                    if(time_free_mode==1)
                    {
                        write_addr_dat_n(0x06,0x20|0x80 , 1);
                    }
                    else
                    write_addr_dat_n(0x06, LCD_ADD[time_1]|0x80, 1);

                }
                    else
                    {
                        if(time_free_mode==1)
                        {
                            write_addr_dat_n(0x06,0x20&0x7f , 1);
                        }
                        else
                        write_addr_dat_n(0x06, LCD_ADD[time_1]&0x7f, 1);

                    }

                     if(Time_Status ==0)
                         {
                             if(Point_Flag==1)
                             {
                                 if(time_free_mode==1)
                                     write_addr_dat_n(0x08, 0x20|0x04, 1);
                                 else
```

```
                                         write_addr_dat_n(0x08,
((((LCD_ADD[time_2]&0xf0)|0x01)&0xf1)&0x7f)|0x04, 1);
                                     }
                                     else
                                     {
                                           if(time_free_mode==1)
                                                 write_addr_dat_n(0x08, 0x20|0x02, 1);
                                           else
                                              write_addr_dat_n(0x08,
(((LCD_ADD[time_2]&0xf0)|0x01)&0xf1)|0x02, 1);
                                     }
                               }
                               else
                               {
                                      if(Point_Flag==1)
                                         {
                                                if(time_free_mode==1)
                                                 write_addr_dat_n(0x08, 0x20|0x04, 1);
                                           else
                                                 write_addr_dat_n(0x08,
((((LCD_ADD[time_2]&0xf0)|0x08)&0xf8)&0x7f)|0x04, 1);
                                         }
                                  else
                                     {
                                           if(time_free_mode==1)
                                                 write_addr_dat_n(0x08, 0x20|0x02, 1);
                                           else
                                                write_addr_dat_n(0x08,
(((LCD_ADD[time_2]&0xf0)|0x08)&0xf8)|0x02, 1);
                                     }
                               }
                  if(time_free_mode==1)
                        write_addr_dat_n(0x0a, 0x0, 1);
                  else
                     write_addr_dat_n(0x0a, (LCD_ADD[time_2]&0x0f)<<4, 1);
      }

      if(Set_Flag2)
      {
            if((Key_Count==0)&&(Key1_Count==0))
      point_add=0;
            else
            point_add=LCD_ADD[time_2];
      }
      else
      point_add=LCD_ADD[time_2];
```

```
}

void speed_deal(void)
{
 if(ADD_Mode==1)
          {
               if(Rpm>Speed_Rel )
                    dis =Speed_Rel;
               else
               {
                    dis=Rpm;
                    ADD_Mode=4;
               }
          }
          else
          {
               if(Rpm<Speed_Rel )
                    dis =Speed_Rel;
               else
               {
                    dis=Rpm;
                    ADD_Mode=4;
               }

          }
}
/
#include "ht1623.h"
/****************************************************************************
*
* ?      ?: delay(uint i)
* ?      ?: 5us??
* ?      ?:
* ? ? ?: ?
*
* ????:
* ??      ??        ??        ???????
*    --------------------------------------------------
*****************************************************************************
/
void delay(uint16_t time)
{
     unsigned char a;
     for(a=100;a>0;a--);


}


void write_mode(unsigned char MODE)     //写入模式,数据 or 命令
```

```
{
        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);
}

/*
*       LCD  命令写入函数
*       入口:cbyte ,控制命令字
*       出口:void
*/
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);
}

/*
*   LCD 地址写入函数
*   入口:cbyte,地址
*   出口:void
*/
void write_address(unsigned char Abyte)
{
    unsigned char i = 0;
    Abyte = Abyte << 1;

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

/*
*   LCD 数据写入函数
*   入口:Dbyte,数据
*   出口:void
*/
void write_data_8bit(unsigned char Dbyte)
{
    int i = 0;

    for (i = 0; i < 8; i++)
    {
        Clr_1625_Wr;
        //Delay_us(10);
        if ((Dbyte >> (7 - i)) & 0x01)
        {
            Set_1625_Dat;
        }
        else
        {
            Clr_1625_Dat;
        }
        delay(10);
        Set_1625_Wr;
        delay(10);
    }
}

void write_data_4bit(unsigned char Dbyte)
{
    int i = 0;

    for (i = 0; i < 4; i++)
    {
        Clr_1625_Wr;
        //Delay_us(10);
        if ((Dbyte >> (3 - i)) & 0x01)
        {
            Set_1625_Dat;
        }
        else
        {
            Clr_1625_Dat;
        }
        delay(10);
        Set_1625_Wr;
        delay(10);
    }
}

//////////////////////////////////////////////////////接口函数
/*
*   LCD  初始化，对 lcd 自身做初始化设置
*   入口:void
*   出口:void
*/
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;
}

/*
*   LCD  清屏函数
*   入口:void
*   出口:void
*/
void lcd_clr(void)
{
    write_addr_dat_n(0x0, 0x00, 50);
}

/*
*   LCD  全显示函数
*   入口:void
*   出口:void
*/
void lcd_all(void)
{
    write_addr_dat_n(0x0, 0xFF,60);
}

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;
}
```