MMS3200 源 程 序

```
#include "main.h"
#include "adc.h"
#include "tim.h"
#include "gpio.h"
void SystemClock_Config(void);
int main(void)
{
   HAL_Init();
   SystemClock_Config();
   MX_GPIO_Init();
   MX_TIM4_Init();
   MX_TIM2_Init();
   MX_TIM3_Init();
   MX_ADC1_Init();
   MX_TIM1_Init();
      System_Init();//系统驱动初始化

   while (1)
   {
         Show_Display();//显示屏幕

   }
}
void SystemClock_Config(void)
{
   RCC_OscInitTypeDef RCC_OscInitStruct = {0};
   RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
   RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};
   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
   RCC_OscInitStruct.HSEState = RCC_HSE_ON;
   RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
   RCC_OscInitStruct.HSIState = RCC_HSI_ON;
   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
   RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
   RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL9;
   if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
   {
      Error_Handler();
   }
   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
   RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

   if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
   {
      Error_Handler();
```

```
  }
  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_ADC;
  PeriphClkInit.AdcClockSelection = RCC_ADCPCLK2_DIV6;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    Error_Handler();
  }
}
void Error_Handler(void)
{
  __disable_irq();
  while (1)
  {
  }
}

void assert_failed(uint8_t *file, uint32_t line)
{

}
#endif

#include "Structs.h"

_sys_ sys;//系统初始化检测

_Work_Num_ Ctrl_Speed;//控制速度（期望值）
_Work_Num_ Set_Speed;//设置速度
_Work_Num_ Display_Speed;//显示速度
_Work_Num_ Rel_Speed;//实际速度
_Work_Num_ Speed;//临时存储速度
_Work_Num_ Speed_New;//用于速度显示处理更新
_Work_Num_ Speed_Last;//用于速度显示处理存储
_Work_Num_ Speed_ADDMode;//用于判断速度时上升还是下降
_Work_Num_ Display_RelSpeed;//用于显示实际速度

_Work_Num_long Ctrl_Time;//控制时间（期望值）
_Work_Num_long Set_Time;//设置时间
_Work_Num_long Display_Time;//显示时间
_Work_Num_long Rel_Time;//实际时间
_Work_Num_long Time;//临时存储值
_Work_Num_Flag SetTime_State;//设置时间的状态
_Work_Num_Flag RelTime_State;//实际时间的状态
_Work_Num_Flag DownTime_Over;//时间倒计时结束
_Work_Num_Flag Speed_Cnt;//输入捕获进入的次数
uint8_t Time_unit;//时间单位

int Set_Temp;//设置温度
int Ctrl_Temp;//控制温度（期望值）
int Ture_Temp;//测得温度
int Rel_Temp;//实际温度
int Temp;//临时存储温度
uint8_t Temp_ADDMode;//温度显示模式
uint8_t Temp_State;//温度的状态
int Temp_New,Temp_Last;//现在温度、之前温度
#include "Speed.h"

/*
*****************************************************************
 * 函数原型：  void Encoder_Init(void)
 * 功      能：  编码器初始化
*****************************************************************
*/
void Encoder_Init(void)
{
	#if (Integration_TYPE == 0)
	HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_1);//motor1  输入捕获
	HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_2);//motor2  输入捕获
	HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_3);//motor3  输入捕获
	HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_4);//motor4  输入捕获
	#elif (Integration_TYPE == 1)
	HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_1);//motor1  输入捕获
	HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_2);//motor2  输入捕获
	HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_3);//motor3  输入捕获
	HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_4);//motor4  输入捕获
	HAL_TIM_IC_Start_IT(&htim1, TIM_CHANNEL_1);//motor5  输入捕获
	HAL_TIM_IC_Start_IT(&htim1, TIM_CHANNEL_2);//motor6  输入捕获
	#elif (Integration_TYPE == 2)
	HAL_TIM_IC_Start_IT(&htim4, TIM_CHANNEL_1);//motor1  输入捕获
	HAL_TIM_IC_Start_IT(&htim4, TIM_CHANNEL_2);//motor2  输入捕获
	HAL_TIM_IC_Start_IT(&htim4, TIM_CHANNEL_3);//motor3  输入捕获
	HAL_TIM_IC_Start_IT(&htim4, TIM_CHANNEL_4);//motor4  输入捕获
	HAL_TIM_IC_Start_IT(&htim1, TIM_CHANNEL_1);//motor5  输入捕获
	HAL_TIM_IC_Start_IT(&htim1, TIM_CHANNEL_2);//motor6  输入捕获
	HAL_TIM_IC_Start_IT(&htim1, TIM_CHANNEL_3);//motor7  输入捕获
	HAL_TIM_IC_Start_IT(&htim1, TIM_CHANNEL_4);//motor8  输入捕获
	#endif
}

/*
*****************************************************************
 * 函数原型：  void Check_Speed(void)
 * 功      能：  检测速度是否停止
*****************************************************************
*/
void Check_Speed(void)
{
	#if (Integration_TYPE == 0)//设置成四联时
	Speed_Cnt.L1++;//每 50ms 进入
	Speed_Cnt.L2++;
	Speed_Cnt.L7++;
	Speed_Cnt.L8++;
	if(Speed_Cnt.L1>=10)//0.5s 发现没出发输入捕获
		Rel_Speed.L1 = 0;//将速度清零
	if(Speed_Cnt.L2>=10)
		Rel_Speed.L2 = 0;
	if(Speed_Cnt.L7>=10)
		Rel_Speed.L7 = 0;
	if(Speed_Cnt.L8>=10)
		Rel_Speed.L8 = 0;
	#elif (Integration_TYPE >= 1)
	Speed_Cnt.L1++;//每 50ms 进入
	Speed_Cnt.L2++;
	Speed_Cnt.L3++;
	Speed_Cnt.L4++;
	Speed_Cnt.L5++;
	Speed_Cnt.L6++;
	#if (Integration_TYPE == 2)//0.5s 发现没出发输入捕获
	Speed_Cnt.L7++;
	Speed_Cnt.L8++;
	if(Speed_Cnt.L7>=10)//0.5s 发现没出发输入捕获
		Rel_Speed.L7 = 0;//将速度清零
	if(Speed_Cnt.L8>=10)
		Rel_Speed.L8 = 0;
	#endif
	if(Speed_Cnt.L1>=10)//0.5s 发现没出发输入捕获
		Rel_Speed.L1 = 0;//将速度清零
	if(Speed_Cnt.L2>=10)
		Rel_Speed.L2 = 0;
	if(Speed_Cnt.L3>=10)
		Rel_Speed.L3 = 0;
	if(Speed_Cnt.L4>=10)
		Rel_Speed.L4 = 0;
	if(Speed_Cnt.L5>=10)
		Rel_Speed.L5 = 0;
	if(Speed_Cnt.L6>=10)
		Rel_Speed.L6 = 0;
	#endif
}

#if (Integration_TYPE == 0)//设置成四联时
/*
*****************************************************************
 *  函数原型：  void TIM3CaptureChannel4Callback(void)
 *  功       能：  Tim3 通道 4 的输入捕获回调函数
*****************************************************************
*/
uint32_t L1_capture,L1_capture1,L1_capture2;
uint32_t rel1;
void TIM3CaptureChannel4Callback(void)
```

```
{
	L1_capture1=__HAL_TIM_GET_COMPARE(&htim3, TIM_CHANNEL_4);//获取 Tim3 通
道 4 的输入捕获
	if(L1_capture1>L1_capture2)
		L1_capture=L1_capture1-L1_capture2;
	else
		L1_capture=L1_capture1+(0xFFFF-L1_capture2);
	if(L1_capture<100)
		return;
	rel1=60000000/(L1_capture*200);//计算速度
	L1_capture2=L1_capture1;
	Rel_Speed.L1=rel1;//将速度赋值给 L1 的实际速度
	Speed_Cnt.L1 = 0;
}

/*
******************************************************************
 * 函数原型：	void TIM3CaptureChannel3Callback(void)
 * 功	能：	Tim3 通道 3 的输入捕获回调函数
******************************************************************
*/
uint32_t L2_capture,L2_capture1,L2_capture2;
uint32_t rel2;
void TIM3CaptureChannel3Callback(void)
{
	L2_capture1=__HAL_TIM_GET_COMPARE(&htim3, TIM_CHANNEL_3);//获取 Tim3 通
道 3 的输入捕获
	if(L2_capture1>L2_capture2)
		L2_capture=L2_capture1-L2_capture2;
	else
		L2_capture=L2_capture1+(0xFFFF-L2_capture2);
	if(L2_capture<100)
		return;
	rel2=60000000/(L2_capture*200);//计算速度
	L2_capture2=L2_capture1;
	Rel_Speed.L2=rel2;//将速度赋值给 L1 的实际速度
	Speed_Cnt.L2 = 0;
}

/*
******************************************************************
 * 函数原型：	void TIM3CaptureChannel2Callback(void)
 * 功	能：	Tim3 通道 2 的输入捕获回调函数
******************************************************************
*/
uint32_t L7_capture,L7_capture1,L7_capture2;
uint32_t rel7;
void TIM3CaptureChannel2Callback(void)
{
	L7_capture1=__HAL_TIM_GET_COMPARE(&htim3, TIM_CHANNEL_2);//获取 Tim3 通
```

```
道 2 的输入捕获
        if(L7_capture1>L7_capture2)
                L7_capture=L7_capture1-L7_capture2;
        else
                L7_capture=L7_capture1+(0xFFFF-L7_capture2);
        if(L7_capture<100)
                return;
        rel7=60000000/(L7_capture*200);//计算速度
        L7_capture2=L7_capture1;
        Rel_Speed.L7=rel7;//将速度赋值给 L7 的实际速度
        Speed_Cnt.L7 = 0;
}

/*
*****************************************************************
  *  函数原型：  void TIM3CaptureChannel1Callback(void)
  *  功      能：  Tim3 通道 1 的输入捕获回调函数
*****************************************************************
*/
uint32_t L8_capture,L8_capture1,L8_capture2;
uint32_t rel8;
void TIM3CaptureChannel1Callback(void)
{
        L8_capture1=__HAL_TIM_GET_COMPARE(&htim3, TIM_CHANNEL_1);//获取 Tim3 通
道 1 的输入捕获
        if(L8_capture1>L8_capture2)
                L8_capture=L8_capture1-L8_capture2;
        else
                L8_capture=L8_capture1+(0xFFFF-L8_capture2);
        if(L8_capture<100)
                return;
        rel8=60000000/(L8_capture*200);//计算速度
        L8_capture2=L8_capture1;
        Rel_Speed.L8=rel8;//将速度赋值给 L8 的实际速度
        Speed_Cnt.L8 = 0;
}
#elif (Integration_TYPE == 1)//设置成六联时
/*
*****************************************************************
  *  函数原型：  void TIM1CaptureChannel1Callback(void)
  *  功      能：  Tim1 通道 1 的输入捕获回调函数
*****************************************************************
*/
uint32_t L1_capture,L1_capture1,L1_capture2;
uint32_t rel1;
void TIM1CaptureChannel1Callback(void)
{
        L1_capture1=__HAL_TIM_GET_COMPARE(&htim1, TIM_CHANNEL_1);//获取 Tim1 通
道 1 的输入捕获
        if(L1_capture1>L1_capture2)
        L1_capture=L1_capture1-L1_capture2;
    else
        L1_capture=L1_capture1+(0xFFFF-L1_capture2);
    if(L1_capture<100)
        return;
    rel1=60000000/(L1_capture*200);//计算速度
    L1_capture2=L1_capture1;
    Rel_Speed.L1=rel1;//将速度赋值给 L1 的实际速度
    Speed_Cnt.L1 = 0;
}

/*
*****************************************************************
 * 函数原型： void TIM1CaptureChannel2Callback(void)
 * 功    能： Tim1 通道 2 的输入捕获回调函数
*****************************************************************
*/
uint32_t L2_capture,L2_capture1,L2_capture2;
uint32_t rel2;
void TIM1CaptureChannel2Callback(void)
{
    L2_capture1=__HAL_TIM_GET_COMPARE(&htim1, TIM_CHANNEL_2);//获取 Tim1 通
道 2 的输入捕获
    if(L2_capture1>L2_capture2)
        L2_capture=L2_capture1-L2_capture2;
    else
        L2_capture=L2_capture1+(0xFFFF-L2_capture2);
    if(L2_capture<100)
        return;
    rel2=60000000/(L2_capture*200);//计算速度
    L2_capture2=L2_capture1;
    Rel_Speed.L2=rel2;//将速度赋值给 L1 的实际速度
    Speed_Cnt.L2 = 0;
}

/*
*****************************************************************
 * 函数原型： void TIM3CaptureChannel4Callback(void)
 * 功    能： Tim3 通道 4 的输入捕获回调函数
*****************************************************************
*/
uint32_t L3_capture, L3_capture1, L3_capture2;
uint32_t rel3;
void TIM3CaptureChannel4Callback()
{
    L3_capture1 = __HAL_TIM_GET_COMPARE(&htim3, TIM_CHANNEL_4);
    if(L3_capture1 > L3_capture2)
        L3_capture = L3_capture1 - L3_capture2;
    else
        L3_capture = L3_capture1 + (0xFFFF - L3_capture2);
        if(L3_capture < 100)
                return;
        rel3 = 60000000 / (L3_capture * 200);
        L3_capture2 = L3_capture1;
        Rel_Speed.L3 = rel3;
        Speed_Cnt.L3 = 0;
}

/*
*****************************************************************
  * 函数原型：  void TIM3CaptureChannel1Callback(void)
  * 功      能：  Tim3 通道 1 的输入捕获回调函数
*****************************************************************
*/
uint32_t L4_capture, L4_capture1, L4_capture2;
uint32_t rel4;
void TIM3CaptureChannel1Callback()
{
        L4_capture1 = __HAL_TIM_GET_COMPARE(&htim3, TIM_CHANNEL_1);
        if(L4_capture1 > L4_capture2)
                L4_capture = L4_capture1 - L4_capture2;
        else
                L4_capture = L4_capture1 + (0xFFFF - L4_capture2);
        if(L4_capture < 100)
                return;
        rel4 = 60000000 / (L4_capture * 200);
        L4_capture2 = L4_capture1;
        Rel_Speed.L4 = rel4;
        Speed_Cnt.L4 = 0;
}

/*
*****************************************************************
  * 函数原型：  void TIM3CaptureChannel3Callback(void)
  * 功      能：  Tim3 通道 3 的输入捕获回调函数
*****************************************************************
*/
uint32_t L5_capture, L5_capture1, L5_capture2;
uint32_t rel5;
void TIM3CaptureChannel3Callback()
{
        L5_capture1 = __HAL_TIM_GET_COMPARE(&htim3, TIM_CHANNEL_3);
        if(L5_capture1 > L5_capture2)
                L5_capture = L5_capture1 - L5_capture2;
        else
                L5_capture = L5_capture1 + (0xFFFF - L5_capture2);
        if(L5_capture < 100)
                return;
        rel5 = 60000000 / (L5_capture * 200);
        L5_capture2 = L5_capture1;
        Rel_Speed.L5 = rel5;
        Speed_Cnt.L5 =0;
}

/*
*****************************************************************
  * 函数原型：  void TIM3CaptureChannel2Callback(void)
  * 功      能：  Tim3 通道 2 的输入捕获回调函数
*****************************************************************
*/
uint32_t L6_capture, L6_capture1, L6_capture2;
uint32_t rel6;
void TIM3CaptureChannel2Callback()
{
        L6_capture1 = __HAL_TIM_GET_COMPARE(&htim3, TIM_CHANNEL_2);
        if(L6_capture1 > L6_capture2)
                L6_capture = L6_capture1 - L6_capture2;
        else
                L6_capture = L6_capture1 + (0xFFFF - L6_capture2);
        if(L6_capture < 100)
                return;
        rel6 = 60000000 / (L6_capture * 200);
        L6_capture2 = L6_capture1;
        Rel_Speed.L6 = rel6;
        Speed_Cnt.L6 =0;
}

#elif (Integration_TYPE == 2)

/*
*****************************************************************
  * 函数原型：  void TIM4CaptureChannel1Callback(void)
  * 功      能：  Tim4 通道 1 的输入捕获回调函数
*****************************************************************
*/
uint32_t L1_capture, L1_capture1, L1_capture2;
uint32_t rel1;
void TIM4CaptureChannel1Callback(void)
{
        L1_capture1 = __HAL_TIM_GET_COMPARE(&htim4, TIM_CHANNEL_1);//获取捕获的
值
        if(L1_capture1 > L1_capture2)//判断是不是比上次多
                L1_capture = L1_capture1 - L1_capture2;//多的话就直接减去上一个值
        else//表示超出满值自动清零后
                L1_capture = L1_capture1 + (0xFFFF - L1_capture2);//将用 65536 满值减去上一个值
        if(L1_capture < 100)//小于 100 的话表示错误
                return;
        rel1 = 60000000 / (L1_capture * 200);//计算转速
        L1_capture2 = L1_capture1;//将当前的值保存
        Rel_Speed.L1 = rel1;//实际值赋值
```

```
	Speed_Cnt.L1 = 0;//将判断标志位清零
}

/*
*****************************************************************
 * 函数原型：	void TIM4CaptureChannel2Callback(void)
 * 功	能：	Tim4 通道 2 的输入捕获回调函数
*****************************************************************
*/
uint32_t L2_capture, L2_capture1, L2_capture2;
uint32_t rel2;
void TIM4CaptureChannel2Callback()
{
	L2_capture1 = __HAL_TIM_GET_COMPARE(&htim4, TIM_CHANNEL_2);
	if(L2_capture1 > L2_capture2)
		L2_capture = L2_capture1 - L2_capture2;
	else
		L2_capture = L2_capture1 + (0xFFFF - L2_capture2);
	if(L2_capture < 100)
		return;
	rel2 = 60000000 / (L2_capture * 200);
	L2_capture2 = L2_capture1;
	Rel_Speed.L2 = rel2;
	Speed_Cnt.L2 = 0;
}

/*
*****************************************************************
 * 函数原型：	void TIM4CaptureChannel3Callback(void)
 * 功	能：	Tim4 通道 3 的输入捕获回调函数
*****************************************************************
*/
uint32_t L3_capture, L3_capture1, L3_capture2;
uint32_t rel3;
void TIM4CaptureChannel3Callback()
{
	L3_capture1 = __HAL_TIM_GET_COMPARE(&htim4, TIM_CHANNEL_3);
	if(L3_capture1 > L3_capture2)
		L3_capture = L3_capture1 - L3_capture2;
	else
		L3_capture = L3_capture1 + (0xFFFF - L3_capture2);
	if(L3_capture < 100)
		return;
	rel3 = 60000000 / (L3_capture * 200);
	L3_capture2 = L3_capture1;
	Rel_Speed.L3 = rel3;
	Speed_Cnt.L3 = 0;
}

/*
```

```
****************************************************************
 * 函数原型：  void TIM4CaptureChannel4Callback(void)
 * 功      能：  Tim4 通道 4 的输入捕获回调函数
****************************************************************
*/
uint32_t L4_capture, L4_capture1, L4_capture2;
uint32_t rel4;
void TIM4CaptureChannel4Callback()
{
	L4_capture1 = __HAL_TIM_GET_COMPARE(&htim4, TIM_CHANNEL_4);
	if(L4_capture1 > L4_capture2)
		L4_capture = L4_capture1 - L4_capture2;
	else
		L4_capture = L4_capture1 + (0xFFFF - L4_capture2);
	if(L4_capture < 100)
		return;
	rel4 = 60000000 / (L4_capture * 200);
	L4_capture2 = L4_capture1;
	Rel_Speed.L4 = rel4;
	Speed_Cnt.L4 = 0;
}

/*
****************************************************************
 * 函数原型：  void TIM1CaptureChannel1Callback(void)
 * 功      能：  Tim1 通道 1 的输入捕获回调函数
****************************************************************
*/
uint32_t L5_capture, L5_capture1, L5_capture2;
uint32_t rel5;
void TIM1CaptureChannel1Callback()
{
	L5_capture1 = __HAL_TIM_GET_COMPARE(&htim1, TIM_CHANNEL_1);
	if(L5_capture1 > L5_capture2)
		L5_capture = L5_capture1 - L5_capture2;
	else
		L5_capture = L5_capture1 + (0xFFFF - L5_capture2);
	if(L5_capture < 100)
		return;
	rel5 = 60000000 / (L5_capture * 200);
	L5_capture2 = L5_capture1;
	Rel_Speed.L5 = rel5;
	Speed_Cnt.L5 =0;
}

/*
****************************************************************
 * 函数原型：  void TIM1CaptureChannel2Callback(void)
 * 功      能：  Tim1 通道 2 的输入捕获回调函数
****************************************************************
```

```
*/
uint32_t L6_capture, L6_capture1, L6_capture2;
uint32_t rel6;
void TIM1CaptureChannel2Callback()
{
	L6_capture1 = __HAL_TIM_GET_COMPARE(&htim1, TIM_CHANNEL_2);
	if(L6_capture1 > L6_capture2)
		L6_capture = L6_capture1 - L6_capture2;
	else
		L6_capture = L6_capture1 + (0xFFFF - L6_capture2);
	if(L6_capture < 100)
		return;
	rel6 = 60000000 / (L6_capture * 200);
	L6_capture2 = L6_capture1;
	Rel_Speed.L6 = rel6;
	Speed_Cnt.L6 =0;
}

/*
*****************************************************************
 * 函数原型： void TIM1CaptureChannel3Callback(void)
 * 功    能： Tim1 通道 3 的输入捕获回调函数
*****************************************************************
*/
uint32_t L7_capture, L7_capture1, L7_capture2;
uint32_t rel7;
void TIM1CaptureChannel3Callback()
{
	L7_capture1 = __HAL_TIM_GET_COMPARE(&htim1, TIM_CHANNEL_3);

	if(L7_capture1 > L7_capture2)
		L7_capture = L7_capture1 - L7_capture2;
	else
		L7_capture = L7_capture1 + (0xFFFF - L7_capture2);
	if(L7_capture < 100)
		return;
	// __HAL_TIM_CLEAR_IT(&htim4, TIM_IT_CC1);
	rel7 = 60000000 / (L7_capture * 200);
	L7_capture2 = L7_capture1;
	Rel_Speed.L7 = rel7;
	Speed_Cnt.L7 =0;
}

/*
*****************************************************************
 * 函数原型： void TIM1CaptureChannel1Cal4back(void)
 * 功    能： Tim1 通道 4 的输入捕获回调函数
*****************************************************************
*/
uint32_t L8_capture, L8_capture1, L8_capture2;
```

```
uint32_t rel8;
void TIM1CaptureChannel4Callback()
{
        L8_capture1 = __HAL_TIM_GET_COMPARE(&htim1, TIM_CHANNEL_4);
        if(L8_capture1 > L8_capture2)
                L8_capture = L8_capture1 - L8_capture2;
        else
                L8_capture = L8_capture1 + (0xFFFF - L8_capture2);
        if(L8_capture < 100)
                return;
        rel8 = 60000000 / (L8_capture * 200);
        L8_capture2 = L8_capture1;
        Rel_Speed.L8 = rel8;
        Speed_Cnt.L8 =0;
}
#endif

/*
*****************************************************************
  * 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(Integration_TYPE == 0)//设置成四联时
        if(htim->Instance == TIM3)
        {

                if(htim->Channel==HAL_TIM_ACTIVE_CHANNEL_1)
                {
                        TIM3CaptureChannel1Callback();
                }
                if(htim->Channel==HAL_TIM_ACTIVE_CHANNEL_2)
                {
                        TIM3CaptureChannel2Callback();
                }
                if(htim->Channel==HAL_TIM_ACTIVE_CHANNEL_3)
                {
                        TIM3CaptureChannel3Callback();
                }
                if(htim->Channel==HAL_TIM_ACTIVE_CHANNEL_4)
                {
                        TIM3CaptureChannel4Callback();
                }
        }
        #elif (Integration_TYPE == 1)//设置成六联时
        if(htim->Instance == TIM1)
        {
                if(htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)
        {
            TIM1CaptureChannel1Callback();
        }
        else if(htim->Channel == HAL_TIM_ACTIVE_CHANNEL_2)
        {
            TIM1CaptureChannel2Callback();
        }
    }
    if(htim->Instance == TIM3)
    {
        if(htim->Channel==HAL_TIM_ACTIVE_CHANNEL_1)
        {
            TIM3CaptureChannel1Callback();
        }
        if(htim->Channel==HAL_TIM_ACTIVE_CHANNEL_2)
        {
            TIM3CaptureChannel2Callback();
        }
        if(htim->Channel==HAL_TIM_ACTIVE_CHANNEL_3)
        {
            TIM3CaptureChannel3Callback();
        }
        if(htim->Channel==HAL_TIM_ACTIVE_CHANNEL_4)
        {
            TIM3CaptureChannel4Callback();
        }
    }
    #elif (Integration_TYPE == 2)//设置成八联时
    if(htim->Instance == TIM1)
    {
        if(htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)
        {
            TIM1CaptureChannel1Callback();
        }
        else if(htim->Channel == HAL_TIM_ACTIVE_CHANNEL_2)
        {
            TIM1CaptureChannel2Callback();
        }
        else if(htim->Channel == HAL_TIM_ACTIVE_CHANNEL_3)
        {
            TIM1CaptureChannel3Callback();
        }
        else
        {
            TIM1CaptureChannel4Callback();
        }
    }
    if(htim->Instance == TIM4)
    {
        if(htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)
```

```
		{
			TIM4CaptureChannel1Callback();
		}
		else if(htim->Channel == HAL_TIM_ACTIVE_CHANNEL_2)
		{
			TIM4CaptureChannel2Callback();
		}
		else if(htim->Channel == HAL_TIM_ACTIVE_CHANNEL_3)
		{
			TIM4CaptureChannel3Callback();
		}
		else
		{
			TIM4CaptureChannel4Callback();
		}
	}
	#endif
}
#include "System_Init.h"

/*
*****************************************************************
  *  函数原型：  void System_Init(void)
  *  功      能：  系统功能初始化
*****************************************************************
*/
void System_Init(void)
{
	/**********系统初始化成功**********/
	sys.Init_ok = 0;

	/**********背光源亮度控制**********/
	#if(Integration_TYPE <= 1)//设置成四联和六联时
	HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_3);
	#elif(Integration_TYPE == 2)//设置成八联时
	HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_1);
	#endif

	/**********加热模块 pem 控制**********/
	#if(Temp_TYPE == 1)//设置成加热款
		#if(Integration_TYPE <= 1)//设置成四联和六联时
		HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_4);
		#elif(Integration_TYPE == 2)//设置成八联时
//		  HAL_TIM_PWM_Start(&htim8, TIM_CHANNEL_1);
		#endif
	#endif

	/**********电机初始化**********/
	Motor_Init();
```

```
    /**********编码器初始化**********/
    Encoder_Init();

    /**********初始化 lcd 屏幕**********/
    Lcd_Init();//初始化
    LCD_Light(LCD_ON);//打卡背光
    Lcd_All();//显示全部内容
    HAL_Delay(1000);//延时 1S
    Lcd_Clr();//清屏

    /**********系统参数初始化**********/
    Work_Option = 1;//工位号 1
    SetALL_int(100,&Display_Speed);//初始化工位显示的速度
    SetALL_int(100,&Set_Speed);//初始化设置速度
    SetALL_int(100,&Ctrl_Speed);//初始化控制速度
    PID_Init();//pid 系数初始化
    Temp_Startup_Check();//温度开机补偿检测
    Beep_Time = 0.1;//蜂鸣器响 0.1S


    /***************系统初始化成功**********/
    sys.Init_ok = 1;
}#include "SetVal.h"

/**********全局变量声明******/
uint8_t SetOK_Flag;//检测是否波动旋钮和设置标志位

/*
*****************************************************************
 * 函数原型：  void Check_Set(void)
 * 功      能：  检测设置
*****************************************************************
*/
void Check_Set(void)
{
    if(EC11A_Knob != 0)
    {
        SetOK_Flag = 1;//检测到波动旋钮，等待退出设置模式
    }
    if(SetOK_Flag == 1)
    {
        if(SetMode_Option == 0)//在设定好后
        {
            Set_Speeds(&Speed,&Set_Speed,&Ctrl_Speed);//比较临时速度，不同就将设置
值赋值
            Set_Speeds(&Speed,&Set_Speed,&Display_Speed);//比较临时速度，不同就将设
置值赋值
            Set_Speeds(&Speed,&Set_Speed,&Speed);//比较临时速度，不同就将设置值赋值

            Set_Times(&Time,&Set_Time,&Rel_Time);//比较临时时间，不同就将设置值赋
```

```
值
            Set_Times(&Time,&Set_Time,&Ctrl_Time);//比较临时时间，不同就将设置值赋
值
            Set_Times(&Time,&Set_Time,&Time);//比较临时时间，不同就将设置值赋值

            if(Temp != Set_Temp)//比较临时温度和设定温度不一样
            {
                Ctrl_Temp = Set_Temp;//将设置温度赋值给控制温度
                Temp = Ctrl_Temp;//将设置温度赋值给临时温度
            }

            if(Temp_ADDMode == 1 && Rel_Temp > Ctrl_Temp)//在加热模式下，显示温度
大于控制温度
            {
                Temp_ADDMode = 0;//重新判断
            }
            else if(Temp_ADDMode == 2 && Rel_Temp < Ctrl_Temp)//在降温模式下，显示
温度小于控制温度
            {
                Temp_ADDMode = 0;//重新判断
            }
            else if(Temp_ADDMode == 3)//在温度状态下
            {
                Temp_ADDMode = 0;//重新判断
            }

            SetOK_Flag = 0;
        }
    }
}

/*
*****************************************************************
 * 函数原型：  void Set_Val(uint8_t flag,uint8_t Work_Option,uint8_t SetMode_Option)
 * 功      能：  设置数值
 * 输      入:   flag ：0 是加  1 是减    Work_Option：工位  SetMode_Option：设置模式
 * 参      数：  uint8_t flag,uint8_t Work_Option,uint8_t SetMode_Option
*****************************************************************
*/
void Set_Val(uint8_t flag,uint8_t Work_Option,uint8_t SetMode_Option)
{
    if(flag == 0)//加
    {
        switch(Work_Option)//工位
        {
            case 1:switch(SetMode_Option)//模式
                   {
                        case 1:Set_Speed.L1 = Set_Speed.L1 + 10;//速度加 10
                               Set_Speed.L1    =    (Set_Speed.L1    <    50)    ?    50    :
Set_Speed.L1;//小于 50 时从 50 开始加
```

```
                                        break;
                                case 2:Set_Temp = Set_Temp + 10;//温度加 1℃
                                        break;
                                case 3:Set_Time.L1 = Set_Time.L1 + 60;//时间加一分钟
                                        break;
                        }break;
                case 2:switch(SetMode_Option)//模式
                        {
                                case 1:Set_Speed.L2 = Set_Speed.L2 + 10;//速度加 10
                                         Set_Speed.L2    =    (Set_Speed.L2    <    50)    ?    50    :
Set_Speed.L2;//小于 50 时从 50 开始加
                                        break;
                                case 2:Set_Temp = Set_Temp + 10;//温度加 1℃
                                        break;
                                case 3:Set_Time.L2 = Set_Time.L2 + 60;//时间加一分钟
                                        break;
                        }break;
                case 3:switch(SetMode_Option)//模式
                        {
                                case 1:Set_Speed.L3 = Set_Speed.L3 + 10;//速度加 10
                                         Set_Speed.L3    =    (Set_Speed.L3    <    50)    ?    50    :
Set_Speed.L3;//小于 50 时从 50 开始加
                                        break;
                                case 2:Set_Temp = Set_Temp + 10;//温度加 1℃
                                        break;
                                case 3:Set_Time.L3 = Set_Time.L3 + 60;//时间加一分钟
                                        break;
                        }break;
                case 4:switch(SetMode_Option)//模式
                        {
                                case 1:Set_Speed.L4 = Set_Speed.L4 + 10;//速度加 10
                                         Set_Speed.L4    =    (Set_Speed.L4    <    50)    ?    50    :
Set_Speed.L4;//小于 50 时从 50 开始加
                                        break;
                                case 2:Set_Temp = Set_Temp + 10;//温度加 1℃
                                        break;
                                case 3:Set_Time.L4 = Set_Time.L4 + 60;//时间加一分钟
                                        break;
                        }break;
                case 5:switch(SetMode_Option)//模式
                        {
                                case 1:Set_Speed.L5 = Set_Speed.L5 + 10;//速度加 10
                                         Set_Speed.L5    =    (Set_Speed.L5    <    50)    ?    50    :
Set_Speed.L5;//小于 50 时从 50 开始加
                                        break;
                                case 2:Set_Temp = Set_Temp + 10;//温度加 1℃
                                        break;
                                case 3:Set_Time.L5 = Set_Time.L5 + 60;//时间加一分钟
                                        break;
                        }break;
                case 6:switch(SetMode_Option)//模式
                         {
                               case 1:Set_Speed.L6 = Set_Speed.L6 + 10;//速度加 10
                                          Set_Speed.L6    =    (Set_Speed.L6    <    50)    ?    50    :
Set_Speed.L6;//小于 50 时从 50 开始加
                                        break;
                               case 2:Set_Temp = Set_Temp + 10;//温度加 1℃
                                        break;
                               case 3:Set_Time.L6 = Set_Time.L6 + 60;//时间加一分钟

                                        break;
                         }break;
                case 7:switch(SetMode_Option)//模式
                         {
                               case 1:Set_Speed.L7 = Set_Speed.L7 + 10;//速度加 10
                                          Set_Speed.L7    =    (Set_Speed.L7    <    50)    ?    50    :
Set_Speed.L7;//小于 50 时从 50 开始加
                                        break;
                               case 2:Set_Temp = Set_Temp + 10;//温度加 1℃

                                        break;
                               case 3:Set_Time.L7 = Set_Time.L7 + 60;//时间加一分钟
                                        break;
                         }break;
                case 8:switch(SetMode_Option)//模式
                         {
                               case 1:Set_Speed.L8 = Set_Speed.L8 + 10;//速度加 10
                                          Set_Speed.L8    =    (Set_Speed.L8    <    50)    ?    50    :
Set_Speed.L8;//小于 50 时从 50 开始加
                                        break;
                               case 2:Set_Temp = Set_Temp + 10;//温度加 1℃
                                        break;
                               case 3:Set_Time.L8 = Set_Time.L8 + 60;//时间加一分钟
                                        break;
                         }break;
           }
      }
      if(flag == 1)
      {
           switch(Work_Option)//工位
           {
                case 1:switch(SetMode_Option)//模式
                         {
                               case 1:Set_Speed.L1 = Set_Speed.L1 - 10;//速度减 10
                                        break;
                               case 2:Set_Temp = Set_Temp - 10;//温度减 1℃
                                        break;
                               case 3:Set_Time.L1 = Set_Time.L1 - 60;//时间减一分钟
                                          break;
                         }break;
```

```
case 2:switch(SetMode_Option)//模式
       {
            case 1:Set_Speed.L2 = Set_Speed.L2 - 10;//速度减 10
                   break;
            case 2:Set_Temp = Set_Temp - 10;//温度减 1℃
                   break;
            case 3:Set_Time.L2 = Set_Time.L2 - 60;//时间减一分钟
                    break;
       }break;
case 3:switch(SetMode_Option)//模式
       {
            case 1:Set_Speed.L3 = Set_Speed.L3 - 10;//速度减 10
                   break;
            case 2:Set_Temp = Set_Temp - 10;//温度减 1℃
                   break;
            case 3:Set_Time.L3 = Set_Time.L3 - 60;//时间减一分钟
                    break;
       }break;
case 4:switch(SetMode_Option)//模式
       {
            case 1:Set_Speed.L4 = Set_Speed.L4 - 10;//速度减 10
                   break;
            case 2:Set_Temp = Set_Temp - 10;//温度减 1℃
                   break;
            case 3:Set_Time.L4 = Set_Time.L4 - 60;//时间减一分钟
                    break;
       }break;
case 5:switch(SetMode_Option)//模式
       {
            case 1:Set_Speed.L5 = Set_Speed.L5 - 10;//速度减 10
                   break;
            case 2:Set_Temp = Set_Temp - 10;//温度减 1℃
                   break;
            case 3:Set_Time.L5 = Set_Time.L5 - 60;//时间减一分钟
                    break;
       }break;
case 6:switch(SetMode_Option)//模式
       {
            case 1:Set_Speed.L6 = Set_Speed.L6 - 10;//速度减 10
                   break;
            case 2:Set_Temp = Set_Temp - 10;//温度减 1℃
                   break;
            case 3:Set_Time.L6 = Set_Time.L6 - 60;//时间减一分钟
                    break;
       }break;
case 7:switch(SetMode_Option)//模式
       {
            case 1:Set_Speed.L7 = Set_Speed.L7 - 10;//速度减 10
                   break;
            case 2:Set_Temp = Set_Temp - 10;//温度减 1℃
```

```
                               break;
                        case 3:Set_Time.L7 = Set_Time.L7 - 60;//时间减一分钟
                                break;
                   }break;
            case 8:switch(SetMode_Option)//模式
                   {
                        case 1:Set_Speed.L8 = Set_Speed.L8 - 10;//速度减 10
                               break;
                        case 2:Set_Temp = Set_Temp - 10;//温度减 1℃
                               break;
                        case 3:Set_Time.L8 = Set_Time.L8 - 60;//时间减一分钟
                                break;
                   }break;
        }
    }
    Set_Speed.L1 = (Set_Speed.L1 > 1500) ? 1500 : Set_Speed.L1;//速度不超过 1500 转
    Set_Speed.L1 = (Set_Speed.L1 < 50) ? 0 : Set_Speed.L1;//速度设置小于 50 转时清零
    Set_Time.L1 = (Set_Time.L1 > 86400) ? 86400 : Set_Time.L1;//时间最多设定 23 小时 59 分钟
    Set_Time.L1 = (Set_Time.L1 < 60) ? 0 : Set_Time.L1;//时间小于 1 分钟不设定
    SetTime_State.L1    = (Set_Time.L1 < 60) ? 0 : 1;//判断是否设置了时间

    Set_Speed.L2 = (Set_Speed.L2 > 1500) ? 1500 : Set_Speed.L2;//速度不超过 1500 转
    Set_Speed.L2 = (Set_Speed.L2 < 50) ? 0 : Set_Speed.L2;//速度设置小于 50 转时清零
    Set_Time.L2 = (Set_Time.L2 > 86400) ? 86400 : Set_Time.L2;//时间最多设定 23 小时 59 分钟
    Set_Time.L2 = (Set_Time.L2 < 60) ? 0 : Set_Time.L2;//时间小于 1 分钟不设定
    SetTime_State.L2    = (Set_Time.L2 < 60) ? 0 : 1;//判断是否设置了时间

    Set_Speed.L3 = (Set_Speed.L3 > 1500) ? 1500 : Set_Speed.L3;//速度不超过 1500 转
    Set_Speed.L3 = (Set_Speed.L3 < 50) ? 0 : Set_Speed.L3;//速度设置小于 50 转时清零
    Set_Time.L3 = (Set_Time.L3 > 86400) ? 86400 : Set_Time.L3;//时间最多设定 23 小时 59 分钟
    Set_Time.L3 = (Set_Time.L3 < 60) ? 0 : Set_Time.L3;//时间小于 1 分钟不设定
    SetTime_State.L3    = (Set_Time.L3 < 60) ? 0 : 1;//判断是否设置了时间

    Set_Speed.L4 = (Set_Speed.L4 > 1500) ? 1500 : Set_Speed.L4;//速度不超过 1500 转
    Set_Speed.L4 = (Set_Speed.L4 < 50) ? 0 : Set_Speed.L4;//速度设置小于 50 转时清零
    Set_Time.L4 = (Set_Time.L4 > 86400) ? 86400 : Set_Time.L4;//时间最多设定 23 小时 59 分钟
    Set_Time.L4 = (Set_Time.L4 < 60) ? 0 : Set_Time.L4;//时间小于 1 分钟不设定
    SetTime_State.L4    = (Set_Time.L4 < 60) ? 0 : 1;//判断是否设置了时间

    Set_Speed.L5 = (Set_Speed.L5 > 1500) ? 1500 : Set_Speed.L5;//速度不超过 1500 转
    Set_Speed.L5 = (Set_Speed.L5 < 50) ? 0 : Set_Speed.L5;//速度设置小于 50 转时清零
    Set_Time.L5 = (Set_Time.L5 > 86400) ? 86400 : Set_Time.L5;//时间最多设定 23 小时 59 分钟
    Set_Time.L5 = (Set_Time.L5 < 60) ? 0 : Set_Time.L5;//时间小于 1 分钟不设定
    SetTime_State.L5    = (Set_Time.L5 < 60) ? 0 : 1;//判断是否设置了时间
    Set_Speed.L6 = (Set_Speed.L6 > 1500) ? 1500 : Set_Speed.L6;//速度不超过 1500 转
    Set_Speed.L6 = (Set_Speed.L6 < 50) ? 0 : Set_Speed.L6;//速度设置小于 50 转时清零
    Set_Time.L6 = (Set_Time.L6 > 86400) ? 86400 : Set_Time.L6;//时间最多设定 23 小时 59 分钟
    Set_Time.L6 = (Set_Time.L6 < 60) ? 0 : Set_Time.L6;//时间小于 1 分钟不设定
    SetTime_State.L6    = (Set_Time.L6 < 60) ? 0 : 1;//判断是否设置了时间

    Set_Speed.L7 = (Set_Speed.L7 > 1500) ? 1500 : Set_Speed.L7;//速度不超过 1500 转
    Set_Speed.L7 = (Set_Speed.L7 < 50) ? 0 : Set_Speed.L7;//速度设置小于 50 转时清零
    Set_Time.L7 = (Set_Time.L7 > 86400) ? 86400 : Set_Time.L7;//时间最多设定 23 小时 59 分钟
    Set_Time.L7 = (Set_Time.L7 < 60) ? 0 : Set_Time.L7;//时间小于 1 分钟不设定
    SetTime_State.L7    = (Set_Time.L7 < 60) ? 0 : 1;//判断是否设置了时间

    Set_Speed.L8 = (Set_Speed.L8 > 1500) ? 1500 : Set_Speed.L8;//速度不超过 1500 转
    Set_Speed.L8 = (Set_Speed.L8 < 50) ? 0 : Set_Speed.L8;//速度设置小于 50 转时清零
    Set_Time.L8 = (Set_Time.L8 > 86400) ? 86400 : Set_Time.L8;//时间最多设定 23 小时 59 分钟
    Set_Time.L8 = (Set_Time.L8 < 60) ? 0 : Set_Time.L8;//时间小于 1 分钟不设定
    SetTime_State.L8    = (Set_Time.L8 < 60) ? 0 : 1;//判断是否设置了时间

    Set_Temp = (Set_Temp > 1200) ? 1200 : Set_Temp;//温度不超过 120℃
    Set_Temp = (Set_Temp < 10) ? 0 : Set_Temp;//温度设置小于 1℃时清零
    Temp_State = (Set_Temp < 10) ? 0 : 1;//判断是否设置了温度

    if(SetMode_Option!=0)//如果在设置模式中转动旋钮
    {
        Twinkle_Time = 6000;//闪烁显示 6S
        EC11A_Knob = 1;//检测是不是在旋动旋钮
        Work_All = 0;//退出同步模式
    }
}

/*
*****************************************************************
 * 函数原型： void SetALL_int(int Val,_Work_Num_ *Work_Num)
 * 功    能： 将结构图中的参数赋值-int 型
 * 输    入: Val 赋予的值   Work_Num：结构体，要用&号连接
 * 参    数： int Val,_Work_Num_ *Work_Num
*****************************************************************
*/
void SetALL_int(int Val,_Work_Num_ *Work_Num)
{
    Work_Num->L1 = Val;
    Work_Num->L2 = Val;
    Work_Num->L3 = Val;
    Work_Num->L4 = Val;
    Work_Num->L5 = Val;
    Work_Num->L6 = Val;
    Work_Num->L7 = Val;
    Work_Num->L8 = Val;
}

/*
*****************************************************************
 * 函数原型： void SetALL_int8(uint8_t Val,_Work_Num_Flag *Work_Num)
 * 功    能： 将结构图中的参数赋值-uint8_t 型
 * 输    入:  Val 赋予的值  Work_Num：结构体，要用&号连接
 * 参    数： uint8_t Val,_Work_Num_Flag *Work_Num
*****************************************************************
*/
void SetALL_int8(uint8_t Val,_Work_Num_Flag *Work_Num)
{
    Work_Num->L1 = Val;
    Work_Num->L2 = Val;
    Work_Num->L3 = Val;
    Work_Num->L4 = Val;
    Work_Num->L5 = Val;
    Work_Num->L6 = Val;
    Work_Num->L7 = Val;
    Work_Num->L8 = Val;
}

/*
*****************************************************************
 * 函数原型： void SetALL_int32(uint32_t Val,_Work_Num_long *Work_Num)
 * 功    能： 将结构图中的参数赋值-uint32_t 型
 * 输    入:  Val 赋予的值  Work_Num：结构体，要用&号连接
 * 参    数： uint32_t Val,_Work_Num_long *Work_Num
*****************************************************************
*/
void SetALL_int32(uint32_t Val,_Work_Num_long *Work_Num)
{
    Work_Num->L1 = Val;
    Work_Num->L2 = Val;
    Work_Num->L3 = Val;
    Work_Num->L4 = Val;
    Work_Num->L5 = Val;
    Work_Num->L6 = Val;
    Work_Num->L7 = Val;
    Work_Num->L8 = Val;
}

/*
*****************************************************************
 * 函数原型： void SetALL_TimeOver(_Work_Num_long *Work_Num1,_Work_Num_long
*Work_Num)
 * 功    能： //将两个结构体变量的参数对应赋值，用于结束时间复原
 * 输    入:  Work_Num1 结构体，要用&号连接  Work_Num：结构体，要用&号连接
 * 参    数： _Work_Num_long *Work_Num1,_Work_Num_long *Work_Num
```

```
*****************************************************************
*/
void SetALL_TimeOver(_Work_Num_long *Work_Num1,_Work_Num_long *Work_Num)
{
    Work_Num1->L1 = Work_Num->L1;
    Work_Num1->L2 = Work_Num->L2;
    Work_Num1->L3 = Work_Num->L3;
    Work_Num1->L4 = Work_Num->L4;
    Work_Num1->L5 = Work_Num->L5;
    Work_Num1->L6 = Work_Num->L6;
    Work_Num1->L7 = Work_Num->L7;
    Work_Num1->L8 = Work_Num->L8;
}

/*
*****************************************************************
 * 函 数 原 型 :   void  SetALL_SpeedOver(_Work_Num_  *Work_Num1,_Work_Num_ *Work_Num)
 * 功    能： 将两个结构体变量的参数对应赋值，用于结束时间速度复原
 * 输    入:  Work_Num1 结构体，要用&号连接  Work_Num：结构体，要用&号连接
 * 参    数： _Work_Num_long *Work_Num1,_Work_Num_long *Work_Num
*****************************************************************
*/
void SetALL_SpeedOver(_Work_Num_ *Work_Num1,_Work_Num_ *Work_Num)
{
    Work_Num1->L1 = Work_Num->L1;
    Work_Num1->L2 = Work_Num->L2;
    Work_Num1->L3 = Work_Num->L3;
    Work_Num1->L4 = Work_Num->L4;
    Work_Num1->L5 = Work_Num->L5;
    Work_Num1->L6 = Work_Num->L6;
    Work_Num1->L7 = Work_Num->L7;
    Work_Num1->L8 = Work_Num->L8;
}
/*
*****************************************************************
 * 函 数 原 型 :  void  Speed_ALL(uint8_t  work,_Work_Num_  *Work_Num,_Work_Num_ *Work_Num1)
 * 功    能： 同步功能，将所有工位的速度同步
 * 输    入:  work 工位号 Work_Num 结构体，要用&号连接  Work_Num1：结构体，要用&号连接
 * 参    数： uint8_t work,_Work_Num_ *Work_Num,_Work_Num_ *Work_Num1
*****************************************************************
*/
void Speed_ALL(uint8_t work,_Work_Num_ *Work_Num,_Work_Num_ *Work_Num1)
{
    switch(work)
    {
        case 1: Work_Num->L1 = Work_Num1->L1;
                Work_Num->L2 = Work_Num1->L1;
                Work_Num->L3 = Work_Num1->L1;
                Work_Num->L4 = Work_Num1->L1;
                Work_Num->L5 = Work_Num1->L1;
                Work_Num->L6 = Work_Num1->L1;
                Work_Num->L7 = Work_Num1->L1;
                Work_Num->L8 = Work_Num1->L1;
                break;
        case 2:Work_Num->L2 = Work_Num1->L2;
                Work_Num->L1 = Work_Num1->L2;
                Work_Num->L3 = Work_Num1->L2;
                Work_Num->L4 = Work_Num1->L2;
                Work_Num->L5 = Work_Num1->L2;
                Work_Num->L6 = Work_Num1->L2;
                Work_Num->L7 = Work_Num1->L2;
                Work_Num->L8 = Work_Num1->L2;
                break;
        case 3:Work_Num->L3 = Work_Num1->L3;
                Work_Num->L1 = Work_Num1->L3;
                Work_Num->L2 = Work_Num1->L3;
                Work_Num->L4 = Work_Num1->L3;
                Work_Num->L5 = Work_Num1->L3;
                Work_Num->L6 = Work_Num1->L3;
                Work_Num->L7 = Work_Num1->L3;
                Work_Num->L8 = Work_Num1->L3;
                break;
        case 4:Work_Num->L4 = Work_Num1->L4;
                Work_Num->L1 = Work_Num1->L4;
                Work_Num->L2 = Work_Num1->L4;
                Work_Num->L3 = Work_Num1->L4;
                Work_Num->L5 = Work_Num1->L4;
                Work_Num->L6 = Work_Num1->L4;
                Work_Num->L7 = Work_Num1->L4;
                Work_Num->L8 = Work_Num1->L4;
                break;
        case 5:
                Work_Num->L5 = Work_Num1->L5;
                Work_Num->L1 = Work_Num1->L5;
                Work_Num->L2 = Work_Num1->L5;
                Work_Num->L3 = Work_Num1->L5;
                Work_Num->L4 = Work_Num1->L5;
                Work_Num->L6 = Work_Num1->L5;
                Work_Num->L7 = Work_Num1->L5;
                Work_Num->L8 = Work_Num1->L5;
                break;
        case 6:Work_Num->L6 = Work_Num1->L6;
                Work_Num->L1 = Work_Num1->L6;
                Work_Num->L2 = Work_Num1->L6;
                Work_Num->L3 = Work_Num1->L6;
                Work_Num->L4 = Work_Num1->L6;
                Work_Num->L5 = Work_Num1->L6;
```

```
                    Work_Num->L7 = Work_Num1->L6;
                    Work_Num->L8 = Work_Num1->L6;
                    break;
          case 7:Work_Num->L7 = Work_Num1->L7;
                    Work_Num->L1 = Work_Num1->L7;
                    Work_Num->L2 = Work_Num1->L7;
                    Work_Num->L3 = Work_Num1->L7;
                    Work_Num->L4 = Work_Num1->L7;
                    Work_Num->L5 = Work_Num1->L7;
                    Work_Num->L6 = Work_Num1->L7;
                    Work_Num->L8 = Work_Num1->L7;
                    break;
          case 8:Work_Num->L8 = Work_Num1->L8;
                    Work_Num->L1 = Work_Num1->L8;
                    Work_Num->L2 = Work_Num1->L8;
                    Work_Num->L3 = Work_Num1->L8;
                    Work_Num->L4 = Work_Num1->L8;
                    Work_Num->L5 = Work_Num1->L8;
                    Work_Num->L6 = Work_Num1->L8;
                    Work_Num->L7 = Work_Num1->L8;
                    break;
     }
}

/*
********************************************************************
 * 函数原型： void Time_ALL(uint8_t work,_Work_Num_long *Work_Num,_Work_Num_long
*Work_Num1)
 * 功      能： 同步功能，将所有工位的时间同步
 * 输      入: work 工位号 Work_Num 结构体，要用&号连接  Work_Num1：结构体，要
用&号连接
 * 参      数： uint8_t work,_Work_Num_long *Work_Num,_Work_Num_long *Work_Num1
********************************************************************
*/
void Time_ALL(uint8_t work,_Work_Num_long *Work_Num,_Work_Num_long *Work_Num1)
{
     switch(work)
     {
          case 1: Work_Num->L1 = Work_Num1->L1;
                    Work_Num->L2 = Work_Num1->L1;
                    Work_Num->L3 = Work_Num1->L1;
                    Work_Num->L4 = Work_Num1->L1;
                    Work_Num->L5 = Work_Num1->L1;
                    Work_Num->L6 = Work_Num1->L1;
                    Work_Num->L7 = Work_Num1->L1;
                    Work_Num->L8 = Work_Num1->L1;
                    break;
          case 2:Work_Num->L2 = Work_Num1->L2;
                    Work_Num->L1 = Work_Num1->L2;
                    Work_Num->L3 = Work_Num1->L2;
```

```
                Work_Num->L4 = Work_Num1->L2;
                Work_Num->L5 = Work_Num1->L2;
                Work_Num->L6 = Work_Num1->L2;
                Work_Num->L7 = Work_Num1->L2;
                Work_Num->L8 = Work_Num1->L2;
                break;
        case 3:Work_Num->L3 = Work_Num1->L3;
                Work_Num->L1 = Work_Num1->L3;
                Work_Num->L2 = Work_Num1->L3;
                Work_Num->L4 = Work_Num1->L3;
                Work_Num->L5 = Work_Num1->L3;
                Work_Num->L6 = Work_Num1->L3;
                Work_Num->L7 = Work_Num1->L3;
                Work_Num->L8 = Work_Num1->L3;
                break;
        case 4:Work_Num->L4 = Work_Num1->L4;
                Work_Num->L1 = Work_Num1->L4;
                Work_Num->L2 = Work_Num1->L4;
                Work_Num->L3 = Work_Num1->L4;
                Work_Num->L5 = Work_Num1->L4;
                Work_Num->L6 = Work_Num1->L4;
                Work_Num->L7 = Work_Num1->L4;
                Work_Num->L8 = Work_Num1->L4;
                break;
        case 5:
                Work_Num->L5 = Work_Num1->L5;
                Work_Num->L1 = Work_Num1->L5;
                Work_Num->L2 = Work_Num1->L5;
                Work_Num->L3 = Work_Num1->L5;
                Work_Num->L4 = Work_Num1->L5;
                Work_Num->L6 = Work_Num1->L5;
                Work_Num->L7 = Work_Num1->L5;
                Work_Num->L8 = Work_Num1->L5;
                break;
        case 6:Work_Num->L6 = Work_Num1->L6;
                Work_Num->L1 = Work_Num1->L6;
                Work_Num->L2 = Work_Num1->L6;
                Work_Num->L3 = Work_Num1->L6;
                Work_Num->L4 = Work_Num1->L6;
                Work_Num->L5 = Work_Num1->L6;
                Work_Num->L7 = Work_Num1->L6;
                Work_Num->L8 = Work_Num1->L6;
                break;
        case 7:Work_Num->L7 = Work_Num1->L7;
                Work_Num->L1 = Work_Num1->L7;
                Work_Num->L2 = Work_Num1->L7;
                Work_Num->L3 = Work_Num1->L7;
                Work_Num->L4 = Work_Num1->L7;
                Work_Num->L5 = Work_Num1->L7;
                Work_Num->L6 = Work_Num1->L7;
```

```
                    Work_Num->L8 = Work_Num1->L7;
                    break;
            case 8:Work_Num->L8 = Work_Num1->L8;
                    Work_Num->L1 = Work_Num1->L8;
                    Work_Num->L2 = Work_Num1->L8;
                    Work_Num->L3 = Work_Num1->L8;
                    Work_Num->L4 = Work_Num1->L8;
                    Work_Num->L5 = Work_Num1->L8;
                    Work_Num->L6 = Work_Num1->L8;
                    Work_Num->L7 = Work_Num1->L8;
                    break;
        }
}

/*
*****************************************************************
 * 函数原型： void Flag_ALL(uint8_t work,_Work_Num_Flag *Work_Num,_Work_Num_Flag
*Work_Num1)
 * 功    能： 同步功能，将所有工位的 flag 同步
 * 输    入:  work 工位号 Work_Num 结构体，要用&号连接  Work_Num1：结构体，要
用&号连接
 * 参    数： uint8_t work,_Work_Num_Flag *Work_Num,_Work_Num_Flag *Work_Num1
*****************************************************************
*/
void Flag_ALL(uint8_t work,_Work_Num_Flag *Work_Num,_Work_Num_Flag *Work_Num1)
{
        switch(work)
        {
            case 1: Work_Num->L1 = Work_Num1->L1;
                    Work_Num->L2 = Work_Num1->L1;
                    Work_Num->L3 = Work_Num1->L1;
                    Work_Num->L4 = Work_Num1->L1;
                    Work_Num->L5 = Work_Num1->L1;
                    Work_Num->L6 = Work_Num1->L1;
                    Work_Num->L7 = Work_Num1->L1;
                    Work_Num->L8 = Work_Num1->L1;
                    break;
            case 2:Work_Num->L2 = Work_Num1->L2;
                    Work_Num->L1 = Work_Num1->L2;
                    Work_Num->L3 = Work_Num1->L2;
                    Work_Num->L4 = Work_Num1->L2;
                    Work_Num->L5 = Work_Num1->L2;
                    Work_Num->L6 = Work_Num1->L2;
                    Work_Num->L7 = Work_Num1->L2;
                    Work_Num->L8 = Work_Num1->L2;
                    break;
            case 3:Work_Num->L3 = Work_Num1->L3;
                    Work_Num->L1 = Work_Num1->L3;
                    Work_Num->L2 = Work_Num1->L3;
                    Work_Num->L4 = Work_Num1->L3;
```

```
                Work_Num->L5 = Work_Num1->L3;
                Work_Num->L6 = Work_Num1->L3;
                Work_Num->L7 = Work_Num1->L3;
                Work_Num->L8 = Work_Num1->L3;
                break;
        case 4:Work_Num->L4 = Work_Num1->L4;
                Work_Num->L1 = Work_Num1->L4;
                Work_Num->L2 = Work_Num1->L4;
                Work_Num->L3 = Work_Num1->L4;
                Work_Num->L5 = Work_Num1->L4;
                Work_Num->L6 = Work_Num1->L4;
                Work_Num->L7 = Work_Num1->L4;
                Work_Num->L8 = Work_Num1->L4;
                break;
        case 5:
                Work_Num->L5 = Work_Num1->L5;
                Work_Num->L1 = Work_Num1->L5;
                Work_Num->L2 = Work_Num1->L5;
                Work_Num->L3 = Work_Num1->L5;
                Work_Num->L4 = Work_Num1->L5;
                Work_Num->L6 = Work_Num1->L5;
                Work_Num->L7 = Work_Num1->L5;
                Work_Num->L8 = Work_Num1->L5;
                break;
        case 6:Work_Num->L6 = Work_Num1->L6;
                Work_Num->L1 = Work_Num1->L6;
                Work_Num->L2 = Work_Num1->L6;
                Work_Num->L3 = Work_Num1->L6;
                Work_Num->L4 = Work_Num1->L6;
                Work_Num->L5 = Work_Num1->L6;
                Work_Num->L7 = Work_Num1->L6;
                Work_Num->L8 = Work_Num1->L6;
                break;
        case 7:Work_Num->L7 = Work_Num1->L7;
                Work_Num->L1 = Work_Num1->L7;
                Work_Num->L2 = Work_Num1->L7;
                Work_Num->L3 = Work_Num1->L7;
                Work_Num->L4 = Work_Num1->L7;
                Work_Num->L5 = Work_Num1->L7;
                Work_Num->L6 = Work_Num1->L7;
                Work_Num->L8 = Work_Num1->L7;
                break;
        case 8:Work_Num->L8 = Work_Num1->L8;
                Work_Num->L1 = Work_Num1->L8;
                Work_Num->L2 = Work_Num1->L8;
                Work_Num->L3 = Work_Num1->L8;
                Work_Num->L4 = Work_Num1->L8;
                Work_Num->L5 = Work_Num1->L8;
                Work_Num->L6 = Work_Num1->L8;
                Work_Num->L7 = Work_Num1->L8;
```

```
                    break;
      }
}

/*
******************************************************************
 *   函 数 原 型 :     void  Set_Speeds(_Work_Num_  *Work_Num,_Work_Num_
*Work_Num1,_Work_Num_ *Work_Num2)
 * 功    能:  判断设置速度数值是否改变
 * 输    入:  Work_Num 临时存储的速度，要用&号连接  Work_Num1：设置的速度，要
用&号连接 Work_Num2：要赋值的速度，要用&号连接
 * 参      数 :   _Work_Num_  *Work_Num,_Work_Num_  *Work_Num1,_Work_Num_
*Work_Num2
******************************************************************
*/
void   Set_Speeds(_Work_Num_    *Work_Num,_Work_Num_    *Work_Num1,_Work_Num_
*Work_Num2)
{
    if(Work_Num->L1 != Work_Num1->L1)
    {
         Work_Num2->L1 = Work_Num1->L1;
         if(Speed_ADDMode.L1  !=  0)//假如工位只有在启动并且设置了速度的情况下不等于
0，不在未处理模式下
               Speed_ADDMode.L1 = 0;//进入未处理，判断加速还是减速
    }
    if(Work_Num->L2 != Work_Num1->L2)
    {
         Work_Num2->L2 = Work_Num1->L2;
         if(Speed_ADDMode.L2  !=  0)//假如工位只有在启动并且设置了速度的情况下不等于
0，不在未处理模式下
                    Speed_ADDMode.L2 = 0;//进入未处理，判断加速还是减速
    }
    if(Work_Num->L3 != Work_Num1->L3)
    {
          Work_Num2->L3 = Work_Num1->L3;
          if(Speed_ADDMode.L3 != 0)//假如工位只有在启动并且设置了速度的情况下不等于
0，不在未处理模式下
               Speed_ADDMode.L3 = 0;//进入未处理，判断加速还是减速
    }
    if(Work_Num->L4 != Work_Num1->L4)
    {
         Work_Num2->L4 = Work_Num1->L4;
         if(Speed_ADDMode.L4  !=  0)//假如工位只有在启动并且设置了速度的情况下不等于
0，不在未处理模式下
                    Speed_ADDMode.L4 = 0;//进入未处理，判断加速还是减速
    }
    if(Work_Num->L5 != Work_Num1->L5)
    {
         Work_Num2->L5 = Work_Num1->L5;
         if(Speed_ADDMode.L5  !=  0)//假如工位只有在启动并且设置了速度的情况下不等于
```

```
0，不在未处理模式下
                Speed_ADDMode.L5 = 0;//进入未处理，判断加速还是减速
   }
   if(Work_Num->L6 != Work_Num1->L6)
   {
        Work_Num2->L6 = Work_Num1->L6;
       if(Speed_ADDMode.L6  !=  0)//假如工位只有在启动并且设置了速度的情况下不等于
0，不在未处理模式下
                Speed_ADDMode.L6 = 0;//进入未处理，判断加速还是减速
   }
   if(Work_Num->L7 != Work_Num1->L7)
   {
        Work_Num2->L7 = Work_Num1->L7;
        if(Speed_ADDMode.L7 != 0)//假如工位只有在启动并且设置了速度的情况下不等于
0，不在未处理模式下
                Speed_ADDMode.L7 = 0;//进入未处理，判断加速还是减速
   }
   if(Work_Num->L8 != Work_Num1->L8)
   {
       Work_Num2->L8 = Work_Num1->L8;
       if(Speed_ADDMode.L8  !=  0)//假如工位只有在启动并且设置了速度的情况下不等于
0，不在未处理模式下
                Speed_ADDMode.L8 = 0;//进入未处理，判断加速还是减速
   }
}

/*
*****************************************************************
 * 函 数 原 型 ：  void  Set_Times(_Work_Num_long  *Work_Num,_Work_Num_long
*Work_Num1,_Work_Num_long *Work_Num2)
 * 功    能： 判断设置时间数值是否改变
 * 输    入:  Work_Num 临时存储的时间，要用&号连接  Work_Num1：设置的时间，要
用&号连接 Work_Num2：要赋值的时间，要用&号连接
 *  参          数  ：   _Work_Num_long  *Work_Num,_Work_Num_long
*Work_Num1,_Work_Num_long *Work_Num2
*****************************************************************
*/
void            Set_Times(_Work_Num_long            *Work_Num,_Work_Num_long
*Work_Num1,_Work_Num_long *Work_Num2)
{
   if(Work_Num->L1 != Work_Num1->L1)
   {
       Work_Num2->L1 = Work_Num1->L1;
       RelTime_State.L1 = SetTime_State.L1;//同步时间状态
       DownTime_Over.L1 = 0;
   }
   if(Work_Num->L2 != Work_Num1->L2)
   {
       Work_Num2->L2 = Work_Num1->L2;
       RelTime_State.L2 = SetTime_State.L2;//同步时间状态
```

```
            DownTime_Over.L2 = 0;
        }
        if(Work_Num->L3 != Work_Num1->L3)
        {
            Work_Num2->L3 = Work_Num1->L3;
             RelTime_State.L3 = SetTime_State.L3;//同步时间状态
            DownTime_Over.L3 = 0;
        }
        if(Work_Num->L4 != Work_Num1->L4)
        {
            Work_Num2->L4 = Work_Num1->L4;
            RelTime_State.L4 = SetTime_State.L4;//同步时间状态
            DownTime_Over.L4 = 0;
        }
        if(Work_Num->L5 != Work_Num1->L5)
        {
            Work_Num2->L5 = Work_Num1->L5;
            RelTime_State.L5 = SetTime_State.L5;//同步时间状态
            DownTime_Over.L5 = 0;
        }
        if(Work_Num->L6 != Work_Num1->L6)
        {
            Work_Num2->L6 = Work_Num1->L6;
            RelTime_State.L6 = SetTime_State.L6;//同步时间状态
            DownTime_Over.L6 = 0;
        }
        if(Work_Num->L7 != Work_Num1->L7)
        {
            Work_Num2->L7 = Work_Num1->L7;
            RelTime_State.L7 = SetTime_State.L7;//同步时间状态
            DownTime_Over.L7 = 0;
        }
        if(Work_Num->L8 != Work_Num1->L8)
        {
            Work_Num2->L8 = Work_Num1->L8;
            RelTime_State.L8 = SetTime_State.L8;//同步时间状态
            DownTime_Over.L8 = 0;
        }
}
```