SMART MOTOR

制作者 Doxygen 1.9.8

1 README

1.1 定时器配置

定时器名	功能	通道号	引脚
TIM1	速度计算定时器	无	无（内部定时器）
TIM2	电机A编码器计数	CH1, CH2	PA0 (CH1), PA1 (CH2)
TIM3	电机PWM驱动	CH3, CH4	PB0 (CH3), PB1 (CH4)
TIM4	电机B编码器计数	CH1, CH2	PB6 (CH1), PB7 (CH2)

1.2 WS2812 颜色映射

值	常量	颜色	RGB值
0	WS2812_BLACK	黑色	(0, 0, 0)
1	WS2812_RED	红色	(255, 0, 0)
2	WS2812_GREEN	绿色	(0, 255, 0)
3	WS2812_BLUE	蓝色	(0, 0, 255)
4	WS2812_YELLOW	黄色	(255, 255, 0)
5	WS2812_PURPLE	紫色	(128, 0, 128)
6	WS2812_CYAN	青色	(0, 255, 255)

1.3 电机宏定义部分

分类	宏定义名称	值	描述
积分限幅	MOTORA_I_MAX	自定义	电机A积分限幅
积分限幅	MOTORB_I_MAX	自定义	电机B积分限幅
PID参数	MOTORA_KP	自定义	电机A PID-KP值
PID参数	MOTORA_KI	自定义	电机A PID-KI值
PID参数	MOTORA_KD	自定义	电机A PID-KD值
PID参数	MOTORB_KP	自定义	电机B PID-KP值
PID参数	MOTORB_KI	自定义	电机B PID-KI值
PID参数	MOTORB_KD	自定义	电机B PID-KD值
电机编号	MOTOR_A	0	电机A编号
电机编号	MOTOR_B	1	电机B编号
运行状态编号	MOTOR_STOP	0	电机停止状态编号
运行状态编号	MOTOR_FORWARD	1	电机正向运行状态编号

分类	宏定义名称	值	描述
运行状态编号	MOTOR_BACKWARD	2	电机反向运行状态编号
安装方向	MOTOR_Clockwise	0	顺时针旋转为正向
安装方向	MOTOR_Anticlockwise	1	逆时针旋转为正向
操作模式	MOTOR_AUTO	0	自动模式（使用内部PID）
操作模式	MOTOR_MANUAL	1	手动模式（使用外部PID）
PWM限幅	MOTOR_PWM_MAX	自定义	PWM最大限幅（补偿后）
PWM限幅	MOTOR_PWM_MIN	自定义	PWM最小限幅（补偿后）
PWM限幅	MOTOR_PWM_Compensate↩_A	自定义	A通道PWM补偿值
PWM限幅	MOTOR_PWM_Compensate↩_B	自定义	B通道PWM补偿值
引脚定义	Direction_TIM_A	自定义	A通道引脚时钟
引脚定义	Direction_Group_A	自定义	A通道引脚组
引脚定义	Direction_AIN1	自定义	A通道方向引脚1
引脚定义	Direction_AIN2	自定义	A通道方向引脚2
引脚定义	Direction_TIM_B	自定义	B通道引脚时钟
引脚定义	Direction_Group_B	自定义	B通道引脚组
引脚定义	Direction_BIN1	自定义	B通道方向引脚1
引脚定义	Direction_BIN2	自定义	B通道方向引脚2

1.4 总连接表

模块	名称	对应引脚	备注
PWM 定时器 - TIM3	PWM↩_A	PB0（CH3）	控制电机 A 的 PWM 信号输出
PWM 定时器 - TIM3	PWM↩_B	PB1（CH4）	控制电机 B 的 PWM 信号输出
编码器定时器 - TIM2	EA-A	PA0	电机 A 编码器 A 相输入
编码器定时器 - TIM2	EA-B	PA1	电机 A 编码器 B 相输入
编码器定时器 - TIM4	EB-A	PB6	电机 B 编码器 A 相输入
编码器定时器 - TIM4	EB-B	PB7	电机 B 编码器 B 相输入
方向控制	AIN1	PA3	控制电机 A 方向的引脚 1
方向控制	AIN2	PA4	控制电机 A 方向的引脚 2
方向控制	BIN1	PA5	控制电机 B 方向的引脚 1
方向控制	BIN2	PA6	控制电机 B 方向的引脚 2
电池检测	ADC	PA2（ADC1 - CH2）	用于检测电池电压
电压电流检测 - INA226	SDA	PB13	与 INA226 通信的数据线
电压电流检测 - INA226	SCL	PB12	与 INA226 通信的时钟线
OLED 显示	SCL	PB5	OLED 显示屏时钟线
OLED 显示	SDA	PB8	OLED 显示屏数据线
OLED 显示	RES	PB9	OLED 显示屏复位引脚
OLED 显示	DC	PB14	OLED 显示屏数据 / 命令控制引脚
OLED 显示	CS	PB15	OLED 显示屏片选引脚
WS2812 LED 灯带	DAT	PA8	控制 WS2812 LED 灯带的引脚
串口传输 - USART1	TX	PA9	USART1 的发送引脚

模块	名称	对应引脚	备注
串口传输 - USART1	RX	PA10	USART1 的接收引脚
按键输入	KEY1	PA7	按键1的接收引脚
按键输入	KEY2	PA11	按键2的接收引脚
按键输入	KEY3	PA12	按键3的接收引脚
蜂鸣器	BEEP	PB3	蜂鸣器的发送引脚

1.5 串口数据包格式

1.5.1 数据包格式:

字段名称	描述
LINK_FrameHeader1	帧头1
LINK_FrameHeader2	帧头2
LINK_CMD	命令字段
LINK_DATA_H	数据高字节
LINK_DATA_L	数据低字节
LINK_CheckSum	校验和字段
SUM	字段总数（枚举值）

1.5.2 发送命令:

常量名称	值	描述
LINK_CMD_IN_NULL	0x00	Null command (no operation)
LINK_CMD_IN_SET_LED	0x01	设置LED，低字节LED编号，高字节LED状态
LINK_CMD_IN_SET_MotorAPWM	0x12	设置电机A PWM（仅手动模式有效）
LINK_CMD_IN_SET_MotorBPWM	0x22	设置电机B PWM（仅手动模式有效）
LINK_CMD_IN_SET_MotorAGoalSpeed	0x13	设置电机A 目标速度（仅自动模式有效）
LINK_CMD_IN_SET_MotorBGoalSpeed	0x23	设置电机B 目标速度（仅自动模式有效）
LINK_CMD_IN_SET_MotorMode	0x04	设置电机模式，1为手动模式，0为自动模式
LINK_CMD_IN_SET_Orientation	0x05	设置模块安装方向，1为逆时针，0为顺时针
LINK_CMD_IN_SET_SwitchOLED	0x06	设置OLED开关，1为开，0为关
LINK_CMD_IN_SET_MotorStatue	0x07	设置电机状态，1为启动，0为停止
LINK_CMD_IN_ASK_ADC	0x08	查询电池电量
LINK_CMD_IN_ASK_MotorASpeed	0x09	查询电机A速度
LINK_CMD_IN_ASK_MotorBSpeed	0x0A	查询电机B速度

2 结构体索引

2.1 结构体

这里列出了所有结构体，并附带简要说明:

3 文件索引

3.1 文件列表

这里列出了所有文档化的文件，并附带简要说明:

4 结构体说明

4.1 FastResponseFilter结构体 参考

成员变量

- float **alpha**
  滤波系数 *(0 < alpha < 1)*
- float **prev_value**
  前一次滤波值
- float **buffer** [3]
  移动平均窗口
- uint8_t **index**
  当前缓冲区索引

该结构体的文档由以下文件生成:

- D:/STM32 Project/STM32F103/smart_motor/USER/HARDWARE/motor.h

4.2 MOTOR_TypeDef结构体 参考

成员变量

- int16_t **PWMvalue**
  *PWM*装载值
- int16_t **Speed**
  电机转速（编码*KPKP*
- float **PID_GoalSpeed**
  目标转速
- float **PID_KP**
  电机*PID-KP*值
- float **PID_KI**
  电机*PID-KI*值
- float **PID_KD**
  电机*PID-KD*值
- float **PID_LastError**
  上一次*ERROR*
- float **PID_Integral**
  当前*ERROR*
- float **PID_I_MAX**
  *PID*积分限幅

该结构体的文档由以下文件生成:

- D:/STM32 Project/STM32F103/smart_motor/USER/HARDWARE/motor.h

5 文件说明

5.1 D:/STM32 Project/STM32F103/smart_motor/USER/HARDWARE/board.c 文件参考

板载按键、LED、蜂鸣器等外设的初始化

```
#include "board.h"
```

函数

- void **Board_Init** (void)
- int **Board_KeyScan** (void)
- void **Board_LED_Toggle** (void)
- void **Board_BEEP_ActiveMs** (int ms)
- void **Board_BEEP_ContinuousActive** (int time)

5.1.1 详细描述

板载按键、LED、蜂鸣器等外设的初始化

作者

maker114

版本

0.1

日期

2025-06-16

注解

- 配置了开发板上的外设以方便使用
- 引脚可以在头文件中更改

5.2 D:/STM32 Project/STM32F103/smart_motor/USER/HARDWARE/board.h 文件参考

板载驱动头文件

```
#include "delay.h"
#include "usart.h"
#include "sys.h"
```

宏定义

- #define **LED** PCout(13)
  *LED*引脚
- #define **KEY1** GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_7)
  *KEY1*引脚
- #define **KEY2** GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_11)
  *KEY2*引脚
- #define **KEY3** GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_12)
  *KEY3*引脚

函数

- void **Board_Init** (void)
- int **Board_KeyScan** (void)
- void **Board_LED_Toggle** (void)
- void **Board_BEEP_ActiveMs** (int ms)
- void **Board_BEEP_ContinuousActive** (int time)

5.2.1 详细描述

板载驱动头文件

作者

maker114

版本

0.1

日期

2025-06-16

5.3 board.h

浏览该文件的文档.

```
00001
00008 #ifndef __BOARD_H
00009 #define __BOARD_H
00010
00011 #include "delay.h"
00012 #include "usart.h"
00013 #include "sys.h"
00014
00015 #define LED PCout(13)
00016 #define KEY1 GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_7)
00017 #define KEY2 GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_11)
00018 #define KEY3 GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_12)
00019
00020 void Board_Init(void);
00021 int Board_KeyScan(void);
00022 void Board_LED_Toggle(void);
00023 void Board_BEEP_ActiveMs(int ms);
00024 void Board_BEEP_ContinuousActive(int time);
00025
00026 #endif
```

5.4 D:/STM32 Project/STM32F103/smart_motor/USER/HARDWARE/motor.c 文件参考

电机驱动函数

```
#include "stm32f10x.h"
#include "motor.h"
#include "board.h"
#include "usart.h"
```

函数

- void **TIM1_UP_IRQHandler** (void)
  定时器*1*中断服务函数
- void **MOTOR_Init** (void)
  总初始化，仅调用一次此函数即可完成全部初始化
- void MOTOR_ADC_Init (void)
  初始化电池电压检测*ADC*
- void MOTOR_GPIO_Init (void)
  电机*GPIO*初始化
- void **MOTOR_CountTIM_Init** (void)
  转速定时器初始化
- void MOTOR_Set_State (int Motor_Channel, int Motor_State)
  选择电机前进方向，一般由内部调用
- void MOTOR_Set_Orientation (uint8_t Orientation)
  设置小车安装方向
- void MOTOR_Set_OperatingMode (uint8_t Mode)
  设置操作模式
- void MOTOR_Set_PWM (uint8_t Motor_channel, int PWM)
  用来设置电机转动速度，自动设置旋转方向
- void MOTOR_Set_PIDGoalSpeed (uint8_t Motor_channel, float Goal_Speed)
  设置*PID*目标速度
- int MOTOR_Get_Speed (int Motor_Channel)
  获取电机转速
- int MOTOR_Get_PWM (int Motor_Channel)
  获取电机*PWM*值
- float MOTOR_Get_BatteryVoltage (void)
  获取电池电压
- float MOTOR_Get_GoalSpeed (uint8_t Motor_Channel)
  获取电机目标速度（自动模式下有效）
- void MOTOR_ENCODER_Init (void)
  初始化电机编码器
- void MOTOR_PWM_Init (void)
  初始化电机驱动*PWM*
- void MOTOR_PWM_Load (uint8_t Motor_Channel, uint16_t PWM)
  装载电机*PWM*
- void **MOTOR_PID_Init** (void)
  初始化*PID*参数
- void MOTOR_PID_Calculate (MOTOR_TypeDef *MotorStructure)
  *PID*计算函数
- void **MOTOR_PID_TimLoop** (void)
  *PID*定时器循环
- void Filter_Init (FastResponseFilter *filter, float alpha, float init_value)
  初始化滤波器*(*内部调用*)*
- int Filter_Process (FastResponseFilter *filter, float raw_value)
  快速响应滤波器处理函数*(*内部调用*)*

变量

- MOTOR_TypeDef **MotorStructure_A**
  *A*通道电机结构体
- MOTOR_TypeDef **MotorStructure_B**
  *B*通道电机结构体
- FastResponseFilter **filter_A**
  *A*通道滤波器
- FastResponseFilter **filter_B**
  *B*通道滤波器

5.4.1 详细描述

电机驱动函数

作者

maker114

版本

0.1

日期

2025-05-20

注解

github仓库详见: SMART MOTOR

5.4.2 函数说明

Filter_Init()

```
void Filter_Init (
            FastResponseFilter * filter,
            float alpha,
            float init_value )
```

初始化滤波器(内部调用)

参数

filter	滤波器实例指针
alpha	滤波系数(建议0.3-0.7)
init_value	初始值

Filter_Process()

```
int Filter_Process (
            FastResponseFilter * filter,
            float raw_value )
```

快速响应滤波器处理函数(内部调用)

参数

filter	滤波器实例指针
raw_value	原始输入值

返回

滤波后的值

MOTOR_ADC_Init()

```
void MOTOR_ADC_Init (
            void  )
```

初始化电池电压检测ADC

注解

通道：ADC1-Channel2
引脚：PA2

MOTOR_ENCODER_Init()

```
void MOTOR_ENCODER_Init (
            void  )
```

初始化电机编码器

注解

- 定时器：TIM2/TIM4
- 引脚：PA0/PA1/PB6/PB7
- 通道映射：
  - MOTOR_A：TIM2
    - PA0->EA-A
    - PA1->EA-B
  - MOTOR_B：TIM4
    - PB6->EB-A
    - PB7->EB-B

MOTOR_Get_BatteryVoltage()

```
float MOTOR_Get_BatteryVoltage (
            void  )
```

获取电池电压

返回

float 电池电压返回值

MOTOR_Get_GoalSpeed()

```
float MOTOR_Get_GoalSpeed (
            uint8_t Motor_Channel )
```

获取电机目标速度（自动模式下有效）

参数

Motor_Channel	电机通道

返回

float PID目标速度

MOTOR_Get_PWM()

```
int MOTOR_Get_PWM (
            int Motor_Channel )
```

获取电机PWM值

参数

Motor_Channel	电机通道

返回

int 返回电机PWM值

- 自动模式下返回PID结果
- 手动模式下返回传入PWM值

MOTOR_Get_Speed()

```
int MOTOR_Get_Speed (
            int Motor_Channel )
```

获取电机转速

参数

Motor_Channel	电机通道

返回

int 返回电机原始转速（正/负）

MOTOR_GPIO_Init()

```
void MOTOR_GPIO_Init (
            void  )
```

电机GPIO初始化

注解

有关引脚配置修改位于motor.h文件中

MOTOR_PID_Calculate()

```
void MOTOR_PID_Calculate (
            MOTOR_TypeDef * MotorStructure )
```

PID计算函数

参数

MotorStructure	电机结构体指针
Goal_Speed	目标速度

MOTOR_PWM_Init()

```
void MOTOR_PWM_Init (
            void  )
```

初始化电机驱动PWM

注解

详细参数

- 频率：10KHz
- 最大比较值：7200

- 通道：TIM3_CH3和TIM3_CH4
  - 引脚：PB0（CH3）和 PB1（CH4）
- 电机映射： PB0->MOTOR_A PB1->MOTOR_B

MOTOR_PWM_Load()

```
void MOTOR_PWM_Load (
            uint8_t Motor_Channel,
            uint16_t PWM )
```

装载电机PWM

参数

Motor_Channel	电机通道（motor_a/motor_b）
PWM	PWM值（0～7200）

注解

- 仅对对应通道的定时器进行占空比装载，不接受负数，不处理方向
- 一般仅内部调用，对电机进行调速需选择MOTOR_Set_PWM函数

MOTOR_Set_OperatingMode()

```
void MOTOR_Set_OperatingMode (
            uint8_t Mode )
```

设置操作模式

参数

Mode	操作模式 • #define MOTOR_MANUAL 0 // 手动模式 • #define MOTOR_AUTO 1 // 自动模式

MOTOR_Set_Orientation()

```
void MOTOR_Set_Orientation (
            uint8_t Orientation )
```

设置小车安装方向

参数

Orientation	旋转方向 • #define MOTOR_Clockwise 0 // 顺时针旋转为正向 • #define MOTOR_Anticlockwise 1 // 逆时针旋转为正向

MOTOR_Set_PIDGoalSpeed()

```
void MOTOR_Set_PIDGoalSpeed (
            uint8_t Motor_channel,
            float Goal_Speed )
```

设置PID目标速度

参数

Motor_channel	电机通道
Goal_Speed	目标速度

MOTOR_Set_PWM()

```
void MOTOR_Set_PWM (
            uint8_t Motor_channel,
            int PWM )
```

用来设置电机转动速度，自动设置旋转方向

参数

Motor_Channel	电机通道
PWM	对应占空比值（MOTOR_PWM_MIN ~ MOTOR_PWM_MAX）

注解

- 设置对应通道的PWM值，根据输入数据的正负与安装方向（MOTOR_Orientation）决定旋转方向
- 为避免PWM过零引脚频繁切换，PWM=0时不会执行MOTOR_Set_State(Motor_channel, MOTOR_↩STOP);

MOTOR_Set_State()

```
void MOTOR_Set_State (
            int Motor_Channel,
            int Motor_State )
```

选择电机前进方向，一般由内部调用

参数

Motor_Channel	电机通道
Motor_State	电机状态 • MOTOR_STOP停止(刹车) • MOTOR_FORWARD前进 • MOTOR_BACKWARD后退

注解

- 运行方向：OTOR_Clockwise顺时针，MOTOR_Anticlockwise逆时针
- 轮子旋转方向与PWM值之间的关系由MOTOR_Orientation决定，即由小车安装与前进方向决定
- 此函数一般仅内部调用，对电机进行调速需选择MOTOR_Set_PWM函数

5.5 D:/STM32 Project/STM32F103/smart_motor/USER/HARDWARE/motor.h 文件参考

电机控制头文件

```
#include "stm32f10x_tim.h"
#include "stm32f10x_rcc.h"
#include "stm32f10x_adc.h"
```

结构体

- struct MOTOR_TypeDef
- struct FastResponseFilter

宏定义

- #define **MOTORA_I_MAX** 4000
  电机*A*积分限幅
- #define **MOTORB_I_MAX** 4000
  电机*B*积分限幅
- #define **MOTORA_KP** -150.0f
  电机*A PID-KP*值
- #define **MOTORA_KI** -3.5f
  电机*A PID-KI*值
- #define **MOTORA_KD** -10.0f
  电机*A PID-KD*值
- #define **MOTORB_KP** -150.0f
  电机*B PID-KP*值
- #define **MOTORB_KI** -3.5f
  电机*B PID-KI*值
- #define **MOTORB_KD** -10.0f
  电机*B PID-KD*值

- #define **MOTOR_A** 0
  电机*A*
- #define **MOTOR_B** 1
  电机*B*
- #define **MOTOR_STOP** 0
  电机停止
- #define **MOTOR_FORWARD** 1
  电机正向
- #define **MOTOR_BACKWARD** 2
  电机反向
- #define **MOTOR_Clockwise** 0
  顺时针旋转为正向
- #define **MOTOR_Anticlockwise** 1
  逆时针旋转为正向
- #define **MOTOR_AUTO** 0
  自动模式（使用内部*PID*）
- #define **MOTOR_MANUAL** 1
  手动模式（使用外部*PID*）
- #define **MOTOR_PWM_MAX** 7000
  *PWM*最大限幅（补偿后）
- #define **MOTOR_PWM_MIN** -7000
  *PWM*最小限幅（补偿后）
- #define **MOTOR_PWM_Compensate_A** 000
  *A*通道*PWM*补偿值
- #define **MOTOR_PWM_Compensate_B** 000
  *B*通道*PWM*补偿值
- #define **Direction_TIM_A** RCC_APB2Periph_GPIOA
  *A*通道引脚时钟
- #define **Direction_Group_A** GPIOA
  *A*通道引脚组
- #define **Direction_AIN1** GPIO_Pin_3
  *A*通道方向引脚*1*
- #define **Direction_AIN2** GPIO_Pin_4
  *A*通道方向引脚*2*
- #define **Direction_TIM_B** RCC_APB2Periph_GPIOA
  *B*通道引脚时钟
- #define **Direction_Group_B** GPIOA
  *B*通道引脚组
- #define **Direction_BIN1** GPIO_Pin_5
  *B*通道方向引脚*1*
- #define **Direction_BIN2** GPIO_Pin_6
  *B*通道方向引脚*2*

函数

- void **MOTOR_Init** (void)
  总初始化，仅调用一次此函数即可完成全部初始化
- void MOTOR_ADC_Init (void)
  初始化电池电压检测*ADC*
- void MOTOR_GPIO_Init (void)

电机*GPIO*初始化

- void **MOTOR_CountTIM_Init** (void)
  转速定时器初始化
- void MOTOR_ENCODER_Init (void)
  初始化电机编码器
- void MOTOR_PWM_Init (void)
  初始化电机驱动*PWM*
- void **MOTOR_PID_Init** (void)
  初始化*PID*参数
- void MOTOR_Set_State (int Motor_Channel, int Motor_State)
  选择电机前进方向，一般由内部调用
- void MOTOR_Set_Orientation (uint8_t Orientation)
  设置小车安装方向
- void MOTOR_Set_OperatingMode (uint8_t Mode)
  设置操作模式
- void MOTOR_Set_PWM (uint8_t Motor_channel, int PWM)
  用来设置电机转动速度，自动设置旋转方向
- void MOTOR_Set_PIDGoalSpeed (uint8_t Motor_channel, float Goal_Speed)
  设置*PID*目标速度
- int MOTOR_Get_Speed (int Motor_Channel)
  获取电机转速
- int MOTOR_Get_PWM (int Motor_Channel)
  获取电机*PWM*值
- float MOTOR_Get_BatteryVoltage (void)
  获取电池电压
- float MOTOR_Get_GoalSpeed (uint8_t Motor_Channel)
  获取电机目标速度（自动模式下有效）
- void MOTOR_PWM_Load (uint8_t Motor_Channel, uint16_t PWM)
  装载电机*PWM*
- void MOTOR_PID_Calculate (MOTOR_TypeDef *MotorStructure)
  *PID*计算函数
- void **MOTOR_PID_TimLoop** (void)
  *PID*定时器循环
- void Filter_Init (FastResponseFilter *filter, float alpha, float init_value)
  初始化滤波器*(*内部调用*)*
- int Filter_Process (FastResponseFilter *filter, float raw_value)
  快速响应滤波器处理函数*(*内部调用*)*

变量

- static uint8_t **MOTOR_Orientation** = MOTOR_Anticlockwise
  根据小车安装与前进方向进行设置
- static uint8_t **MOTOR_OperatingMode** = MOTOR_AUTO
  根据实际需求进行设置，默认为自动模式

5.5.1 详细描述

电机控制头文件

作者

maker114

版本

0.1

日期

2025-06-16

注解

宏定义部分详见README.md

5.5.2 函数说明

Filter_Init()

```
void Filter_Init (
            FastResponseFilter * filter,
            float alpha,
            float init_value )
```

初始化滤波器(内部调用)

参数

filter	滤波器实例指针
alpha	滤波系数(建议0.3-0.7)
init_value	初始值

Filter_Process()

```
int Filter_Process (
            FastResponseFilter * filter,
            float raw_value )
```

快速响应滤波器处理函数(内部调用)

参数

filter	滤波器实例指针
raw_value	原始输入值

返回

滤波后的值

MOTOR_ADC_Init()

```
void MOTOR_ADC_Init (
            void  )
```

初始化电池电压检测ADC

注解

通道：ADC1-Channel2
引脚：PA2

MOTOR_ENCODER_Init()

```
void MOTOR_ENCODER_Init (
            void  )
```

初始化电机编码器

注解

- 定时器：TIM2/TIM4
- 引脚：PA0/PA1/PB6/PB7
- 通道映射：
  - MOTOR_A：TIM2
    - PA0->EA-A
    - PA1->EA-B
  - MOTOR_B：TIM4
    - PB6->EB-A
    - PB7->EB-B

MOTOR_Get_BatteryVoltage()

```
float MOTOR_Get_BatteryVoltage (
            void  )
```

获取电池电压

返回

float 电池电压返回值

MOTOR_Get_GoalSpeed()

```
float MOTOR_Get_GoalSpeed (
            uint8_t Motor_Channel )
```

获取电机目标速度（自动模式下有效）

参数

Motor_Channel	电机通道

返回

float PID目标速度

MOTOR_Get_PWM()

```
int MOTOR_Get_PWM (
            int Motor_Channel )
```

获取电机PWM值

参数

Motor_Channel	电机通道

返回

int 返回电机PWM值

- 自动模式下返回PID结果
- 手动模式下返回传入PWM值

MOTOR_Get_Speed()

```
int MOTOR_Get_Speed (
            int Motor_Channel )
```

获取电机转速

参数

Motor_Channel	电机通道

返回

int 返回电机原始转速（正/负）

MOTOR_GPIO_Init()

```
void MOTOR_GPIO_Init (
            void  )
```

电机GPIO初始化

注解

有关引脚配置修改位于motor.h文件中

MOTOR_PID_Calculate()

```
void MOTOR_PID_Calculate (
            MOTOR_TypeDef * MotorStructure )
```

PID计算函数

参数

MotorStructure	电机结构体指针
Goal_Speed	目标速度

MOTOR_PWM_Init()

```
void MOTOR_PWM_Init (
            void  )
```

初始化电机驱动PWM

注解

详细参数

- 频率：10KHz
- 最大比较值：7200

- 通道：TIM3_CH3和TIM3_CH4
  - 引脚：PB0（CH3）和 PB1（CH4）
- 电机映射： PB0->MOTOR_A PB1->MOTOR_B

MOTOR_PWM_Load()

```
void MOTOR_PWM_Load (
            uint8_t Motor_Channel,
            uint16_t PWM )
```

装载电机PWM

参数

Motor_Channel	电机通道（motor_a/motor_b）
PWM	PWM值（0~7200）

注解

- 仅对对应通道的定时器进行占空比装载，不接受负数，不处理方向
- 一般仅内部调用，对电机进行调速需选择MOTOR_Set_PWM函数

MOTOR_Set_OperatingMode()

```
void MOTOR_Set_OperatingMode (
            uint8_t Mode )
```

设置操作模式

参数

Mode	操作模式 • #define MOTOR_MANUAL 0 // 手动模式 • #define MOTOR_AUTO 1 // 自动模式

MOTOR_Set_Orientation()

```
void MOTOR_Set_Orientation (
            uint8_t Orientation )
```

设置小车安装方向

参数

Orientation	旋转方向 • #define MOTOR_Clockwise 0 // 顺时针旋转为正向 • #define MOTOR_Anticlockwise 1 // 逆时针旋转为正向

MOTOR_Set_PIDGoalSpeed()

```
void MOTOR_Set_PIDGoalSpeed (
            uint8_t Motor_channel,
            float Goal_Speed )
```

设置PID目标速度

参数

Motor_channel	电机通道
Goal_Speed	目标速度

MOTOR_Set_PWM()

```
void MOTOR_Set_PWM (
            uint8_t Motor_channel,
            int PWM )
```

用来设置电机转动速度，自动设置旋转方向

参数

Motor_Channel	电机通道
PWM	对应占空比值（MOTOR_PWM_MIN ~ MOTOR_PWM_MAX）

注解

- 设置对应通道的PWM值，根据输入数据的正负与安装方向（MOTOR_Orientation）决定旋转方向
- 为避免PWM过零引脚频繁切换，PWM=0时不会执行MOTOR_Set_State(Motor_channel, MOTOR_STOP);

MOTOR_Set_State()

```
void MOTOR_Set_State (
            int Motor_Channel,
            int Motor_State )
```

选择电机前进方向，一般由内部调用

参数

Motor_Channel	电机通道
Motor_State	电机状态 • MOTOR_STOP停止(刹车) • MOTOR_FORWARD前进 • MOTOR_BACKWARD后退

注解

- 运行方向：OTOR_Clockwise顺时针，MOTOR_Anticlockwise逆时针
- 轮子旋转方向与PWM值之间的关系由MOTOR_Orientation决定，即由小车安装与前进方向决定
- 此函数一般仅内部调用，对电机进行调速需选择MOTOR_Set_PWM函数

5.6 motor.h

浏览该文件的文档.

```
00001
00009 #ifndef __MOTOR_H
00010 #define __MOTOR_H
00011
00012 #include "stm32f10x_tim.h"
00013 #include "stm32f10x_rcc.h"
00014 #include "stm32f10x_adc.h"
00015 /********************配置部分**********************/
00016 // 电机积分限幅
00017 #define MOTORA_I_MAX 4000
00018 #define MOTORB_I_MAX 4000
00019 // 电机PID
00020 #define MOTORA_KP -150.0f
00021 #define MOTORA_KI -3.5f
00022 #define MOTORA_KD -10.0f
00023
00024 #define MOTORB_KP -150.0f
00025 #define MOTORB_KI -3.5f
00026 #define MOTORB_KD -10.0f
00027 // 电机编号
00028 #define MOTOR_A 0
00029 #define MOTOR_B 1
00030 // 运行状态编号
00031 #define MOTOR_STOP 0
00032 #define MOTOR_FORWARD 1
00033 #define MOTOR_BACKWARD 2
00034 //  安装方向
00035 #define MOTOR_Clockwise 0
00036 #define MOTOR_Anticlockwise 1
00037 // 操作模式
00038 #define MOTOR_AUTO 0
00039 #define MOTOR_MANUAL 1
00040 // PWM限幅（可调范围-7199~+7199）
00041 #define MOTOR_PWM_MAX 7000
00042 #define MOTOR_PWM_MIN -7000
00043 #define MOTOR_PWM_Compensate_A 000
00044 #define MOTOR_PWM_Compensate_B 000
00045 //  通道A引脚
00046 #define Direction_TIM_A RCC_APB2Periph_GPIOA
00047 #define Direction_Group_A GPIOA
00048 #define Direction_AIN1 GPIO_Pin_3
00049 #define Direction_AIN2 GPIO_Pin_4
00050 // 通道B引脚
00051 #define Direction_TIM_B RCC_APB2Periph_GPIOA
00052 #define Direction_Group_B GPIOA
00053 #define Direction_BIN1 GPIO_Pin_5
00054 #define Direction_BIN2 GPIO_Pin_6
00055
00056 /********************可变设置部分**********************/
00057 // 安装方向
00058 static uint8_t MOTOR_Orientation = MOTOR_Anticlockwise;
00059 // 操作模式
00060 static uint8_t MOTOR_OperatingMode = MOTOR_AUTO;
00061 /********************结构体部分**********************/
00062 typedef struct
00063 {
00064     int16_t PWMvalue;
00065     int16_t Speed;
00066     float PID_GoalSpeed;
00067     float PID_KP;
00068     float PID_KI;
00069     float PID_KD;
00070     float PID_LastError;
00071     float PID_Integral;
00072     float PID_I_MAX;
00073 } MOTOR_TypeDef;
00074
00075 typedef struct
00076 {
00077     float alpha;
00078     float prev_value;
00079     float buffer[3];
00080     uint8_t index;
00081 } FastResponseFilter;
00082
00083 /********************函数部分**********************/
00084 // 初始化函数
00085 void MOTOR_Init(void);           // 电机模块全局初始化
00086 void MOTOR_ADC_Init(void);       // 电机相关ADC初始化
00087 void MOTOR_GPIO_Init(void);      // 电机控制GPIO引脚初始化
00088 void MOTOR_CountTIM_Init(void); // 电机编码器计数定时器初始化
00089 void MOTOR_ENCODER_Init(void);  // 电机编码器接口初始化
00090 void MOTOR_PWM_Init(void);       // 电机PWM输出定时器初始化
00091 void MOTOR_PID_Init(void);       // 电机PID控制器参数初始化
00092 // 电机控制设置函数
00093 void MOTOR_Set_State(int Motor_Channel, int Motor_State);             // 设置电机状态
00094 void MOTOR_Set_Orientation(uint8_t Orientation);                      // 设置电机旋转方向
```

```
00095 void MOTOR_Set_OperatingMode(uint8_t Mode);                              // 设置电机工作模式
00096 void MOTOR_Set_PWM(uint8_t Motor_channel, int PWM);                       // 直接设置电机PWM占空比
00097 void MOTOR_Set_PIDGoalSpeed(uint8_t Motor_channel, float Goal_Speed); // 设置PID目标转速
00098 // 状态获取函数
00099 int MOTOR_Get_Speed(int Motor_Channel);            // 获取电机当前实际转速
00100 int MOTOR_Get_PWM(int Motor_Channel);              // 获取电机当前PWM占空比值
00101 float MOTOR_Get_BatteryVoltage(void);              // 获取电机供电电池电压
00102 float MOTOR_Get_GoalSpeed(uint8_t Motor_Channel); // 获取电机PID目标转速设定值
00103 // PWM驱动函数
00104 void MOTOR_PWM_Load(uint8_t Motor_Channel, uint16_t PWM); // 写入PWM值到硬件寄存器（底层驱动）
00105 // PID控制函数
00106 void MOTOR_PID_Calculate(MOTOR_TypeDef *MotorStructure); // 执行PID计算并更新控制量
00107 void MOTOR_PID_TimLoop(void);                             // PID定时控制循环（周期调用）
00108 // 滤波函数
00109 void Filter_Init(FastResponseFilter *filter, float alpha, float init_value); // 初始化快速响应滤波器
00110 int Filter_Process(FastResponseFilter *filter, float raw_value);             // 滤波处理输入数据
00111 #endif
```

5.7 D:/STM32 Project/STM32F103/smart_motor/USER/HARDWARE/usart_link.c 文件参考

串口通信协议程序

```
#include "usart_link.h"
#include "WS2812.h"
#include "motor.h"
#include "oled.h"
#include "board.h"
#include "usart.h"
```

函数

- void LINK_Init (void)
  初始化串口连接
- void LINK_SendPack (uint8_t CMD, uint16_t DATA)
  发送数据包函数
- void **USART3_IRQHandler** (void)
  串口中断处理函数
- void LINK_GetPack (void)
  从*USART3*接收数据包并进行处理。
- void LINK_HandleData (uint8_t CMD, int16_t DATA)
  处理接收到的数据命令。

变量

- uint8_t **packet** [SUM]
- uint8_t **GetPack_Buffer** [SUM]
- uint8_t **GetPack_DATA** [SUM]

5.7.1 详细描述

串口通信协议程序

作者

maker114

版本

0.1

日期

2025-05-22

5.7.2 函数说明

LINK_GetPack()

```
void LINK_GetPack (
            void  )
```

从USART3接收数据包并进行处理。

注解

该函数通过USART3接收一个数据包，包括帧头、命令、数据和高低字节，并进行校验。如果校验成功，则处理数据；否则，丢弃数据包。

LINK_HandleData()

```
void LINK_HandleData (
            uint8_t CMD,
            int16_t DATA )
```

处理接收到的数据命令。

注解

根据接收到的命令字节（CMD）和数据（DATA），执行相应的操作。

参数

CMD	命令字节，指定要执行的操作。
DATA	数据，与命令相关联的具体数据。

LINK_Init()

```
void LINK_Init (
            void  )
```

初始化串口连接

注解

使用串口三接收

LINK_SendPack()

```
void LINK_SendPack (
            uint8_t CMD,
            uint16_t DATA )
```

发送数据包函数

参数

CMD	命令字节
DATA	16位数据

5.8 D:/STM32 Project/STM32F103/smart_motor/USER/HARDWARE/usart_link.h 文件参考

串口通信协议程序头文件

```
#include "stm32f10x.h"
#include "stm32f10x_usart.h"
```

宏定义

- #define **LINK_FRAME_HEADER1** 0xA1
  帧头*1*
- #define **LINK_FRAME_HEADER2** 0xB1
  帧头*2*

枚举

- enum linkpackenum {
  LINK_FrameHeader1 , LINK_FrameHeader2 , LINK_CMD , LINK_DATA_H ,
  LINK_DATA_L , LINK_CheckSum , SUM }

- enum linkcmd_in { **LINK_CMD_IN_NULL** = 0x00 , LINK_CMD_IN_SET_LED = 0x01 , LINK_CMD_IN_SET_MotorAPWM = 0x12 , LINK_CMD_IN_SET_MotorBPWM = 0x22 , LINK_CMD_IN_SET_MotorAGoalSpeed = 0x13 , LINK_CMD_IN_SET_MotorBGoalSpeed = 0x23 , LINK_CMD_IN_SET_MotorMode = 0x04 , LINK_CMD_IN_SET_Orientation = 0x05 , LINK_CMD_IN_SET_SwitchOLED = 0x06 , LINK_CMD_IN_SET_MotorStatue = 0x07 , LINK_CMD_IN_ASK_ADC = 0x08 , LINK_CMD_IN_ASK_MotorASpeed = 0x09 , LINK_CMD_IN_ASK_MotorBSpeed = 0x0A }
- enum linkcmd_out { LINK_CMD_OUT_NULL = 0x00 , LINK_CMD_OUT_RE_ADC = 0x01 , LINK_CMD_OUT_RE_MotorASpeed = 0x02 , LINK_CMD_OUT_RE_MotorBSpeed = 0x03 }

函数

- void LINK_Init (void)
  初始化串口连接
- void LINK_SendPack (uint8_t CMD, uint16_t DATA)
  发送数据包函数
- void LINK_GetPack (void)
  从*USART3*接收数据包并进行处理。
- void LINK_HandleData (uint8_t CMD, int16_t DATA)
  处理接收到的数据命令。

5.8.1 详细描述

串口通信协议程序头文件

作者

maker114

版本

0.1

日期

2025-06-16

5.8.2 枚举类型说明

linkcmd_in

```
enum linkcmd_in
```

枚举值

LINK_CMD_IN_SET_LED	设置LED，低字节LED编号，高字节LED状态
LINK_CMD_IN_SET_MotorAPWM	设置电机A PWM（仅手动模式有效）
LINK_CMD_IN_SET_MotorBPWM	设置电机B PWM（仅手动模式有效）

枚举值

LINK_CMD_IN_SET_MotorAGoalSpeed	设置电机A 目标速度（仅自动模式有效）
LINK_CMD_IN_SET_MotorBGoalSpeed	设置电机B 目标速度（仅自动模式有效）
LINK_CMD_IN_SET_MotorMode	设置电机模式，1为手动模式，0为自动模式
LINK_CMD_IN_SET_Orientation	设置模块安装方向，1为逆时针，0为顺时针
LINK_CMD_IN_SET_SwitchOLED	设置OLED开关，1为开，0为关
LINK_CMD_IN_SET_MotorStatue	设置电机状态，1为启动，0为停止
LINK_CMD_IN_ASK_ADC	查询电池电量
LINK_CMD_IN_ASK_MotorASpeed	查询电机A速度
LINK_CMD_IN_ASK_MotorBSpeed	查询电机B速度

linkcmd_out

```
enum linkcmd_out
```

枚举值

LINK_CMD_OUT_NULL	无
LINK_CMD_OUT_RE_ADC	回传电池电量(倍率=100)
LINK_CMD_OUT_RE_MotorASpeed	回传电机A速度(倍率=1)
LINK_CMD_OUT_RE_MotorBSpeed	回传电机B速度(倍率=1)

linkpackenum

```
enum linkpackenum
```

枚举值

LINK_FrameHeader1	帧头1
LINK_FrameHeader2	帧头2
LINK_CMD	指令
LINK_DATA_H	数据高字节
LINK_DATA_L	数据低字节
LINK_CheckSum	校验和
SUM	数据包长度

5.8.3 函数说明

LINK_GetPack()

```
void LINK_GetPack (
            void  )
```

从USART3接收数据包并进行处理。

注解

该函数通过USART3接收一个数据包，包括帧头、命令、数据和高低字节，并进行校验。如果校验成功，则处理数据；否则，丢弃数据包。

LINK_HandleData()

```
void LINK_HandleData (
            uint8_t CMD,
            int16_t DATA )
```

处理接收到的数据命令。

注解

根据接收到的命令字节（CMD）和数据（DATA），执行相应的操作。

参数

CMD	命令字节，指定要执行的操作。
DATA	数据，与命令相关联的具体数据。

LINK_Init()

```
void LINK_Init (
            void  )
```

初始化串口连接

注解

使用串口三接收

LINK_SendPack()

```
void LINK_SendPack (
            uint8_t CMD,
            uint16_t DATA )
```

发送数据包函数

参数

CMD	命令字节
DATA	16位数据

5.9 usart_link.h

浏览该文件的文档.

```
00001
00008 #ifndef __USARTLINK_H
00009 #define __USARTLINK_H
00010
00011 #include "stm32f10x.h"
00012 #include "stm32f10x_usart.h"
00013
00014 #define LINK_FRAME_HEADER1 0xA1
00015 #define LINK_FRAME_HEADER2 0xB1
00016
00017 typedef enum
00018 {
00019     LINK_FrameHeader1,
00020     LINK_FrameHeader2,
00021     LINK_CMD,
00022     LINK_DATA_H,
00023     LINK_DATA_L,
00024     LINK_CheckSum,
00025     SUM
00026 } linkpackenum;
00027
00028 typedef enum
00029 {
00030     LINK_CMD_IN_NULL = 0x00,
00031     LINK_CMD_IN_SET_LED = 0x01,
00032     LINK_CMD_IN_SET_MotorAPWM = 0x12,
00033     LINK_CMD_IN_SET_MotorBPWM = 0x22,
00034     LINK_CMD_IN_SET_MotorAGoalSpeed = 0x13,
00035     LINK_CMD_IN_SET_MotorBGoalSpeed = 0x23,
00036     LINK_CMD_IN_SET_MotorMode = 0x04,
00037     LINK_CMD_IN_SET_Orientation = 0x05,
00038     LINK_CMD_IN_SET_SwitchOLED = 0x06,
00039     LINK_CMD_IN_SET_MotorStatue = 0x07,
00040     LINK_CMD_IN_ASK_ADC = 0x08,
00041     LINK_CMD_IN_ASK_MotorASpeed = 0x09,
00042     LINK_CMD_IN_ASK_MotorBSpeed = 0x0A,
00043 } linkcmd_in;
00044
00045 typedef enum
00046 {
00047     LINK_CMD_OUT_NULL = 0x00,
00048     LINK_CMD_OUT_RE_ADC = 0x01,
00049     LINK_CMD_OUT_RE_MotorASpeed = 0x02,
00050     LINK_CMD_OUT_RE_MotorBSpeed = 0x03,
00051 } linkcmd_out;
00052
00053 void LINK_Init(void);
00054 void LINK_SendPack(uint8_t CMD, uint16_t DATA);
00055 void LINK_GetPack(void);
00056 void LINK_HandleData(uint8_t CMD, int16_t DATA);
00057
00058 #endif
```

5.10 D:/STM32 Project/STM32F103/smart_motor/USER/HARDWARE/WS2812.c 文件参考

WS2812驱动程序

```
#include "WS2812.h"
```

函数

- void WS2812_init (uint8_t NUM)

  初始化*WS2812*引脚并定义灯的数量
- void ws281x_delay (unsigned int delay_num)

  *ws281x*模块用到的延时函数
- void **ws281x_sendLow** (void)

  根据*WS281x*芯片时序图编写的发送*0*码，*1*码与*RESET*码的函数

- void **ws281x_sendHigh** (void)
  发送*1*码
- void **ws2811_Reset** (void)
  发送*RESET*码
- void ws281x_sendOne (uint32_t dat)
  发送点亮一个灯的数据（即*24bit*）
- void WS2812_SendColor (uint8_t NUM, uint8_t R, uint8_t G, uint8_t B)
  以*RGB*的形式发送颜色
- void WS2812_SendColor_u32 (uint8_t NUM, uint32_t Color)
  以*24*位*GRB*的形式发送颜色
- uint32_t WS2812_Wheel (uint32_t pos)
  色彩轮盘（*256*色）
- void WS2812_StreamColor (u8 pos)
  让所有灯珠显示以*pos*为起始的颜色流，在色轮上以步长*step*递增

变量

- int **LED_NUM** = 0
- uint32_t **LED_BUFFER** [100] = {0}

5.10.1 详细描述

WS2812驱动程序

作者

maker114

版本

0.1

日期

2025-06-16

5.10.2 函数说明

WS2812_init()

```
void WS2812_init (
            uint8_t NUM )
```

初始化WS2812引脚并定义灯的数量

参数

NUM	LED灯数量

WS2812_SendColor()

```
void WS2812_SendColor (
             uint8_t NUM,
             uint8_t R,
             uint8_t G,
             uint8_t B )
```

以RGB的形式发送颜色

参数

NUM	对应灯珠的编号，从0开始
R	R通道颜色值
G	G通道颜色值
B	B通道颜色值

WS2812_SendColor_u32()

```
void WS2812_SendColor_u32 (
             uint8_t NUM,
             uint32_t Color )
```

以24位GRB的形式发送颜色

参数

NUM	对应灯珠的编号，从0开始
Color	颜色的GRB编码（24位）

注解

常常搭配WS2812_Wheel函数使用

WS2812_StreamColor()

```
void WS2812_StreamColor (
             u8 pos )
```

让所有灯珠显示以pos为起始的颜色流，在色轮上以步长step递增

参数

pos	起始颜色

WS2812_Wheel()

```
uint32_t WS2812_Wheel (
            uint32_t pos )
```

色彩轮盘（256色）

参数

pos	颜色值

返回

uint32_t 输出的GRB编码，搭配WS2812_SendColor_u32函数使用

ws281x_delay()

```
void ws281x_delay (
            unsigned int delay_num )
```

ws281x模块用到的延时函数

参数

delay_num	:延时数 （示波器测量延时时间 = delay_num ∗ 440ns ）

ws281x_sendOne()

```
void ws281x_sendOne (
            uint32_t dat )
```

发送点亮一个灯的数据（即24bit）

参数

dat: 颜色的*24*位编码	

注解

不包含帧间延时

5.11 D:/STM32 Project/STM32F103/smart_motor/USER/HARDWARE/WS2812.h 文件参考

WS2812驱动头文件

```
#include "delay.h"
#include "usart.h"
#include "sys.h"
```

宏定义

- #define **WS2812_BLACK** 0x000000
  黑色
- #define **WS2812_WHITE** 0xFFFFFF
  白色
- #define **WS2812_RED** 0x00FF00
  红色
- #define **WS2812_GREEN** 0xFF0000
  绿色
- #define **WS2812_BLUE** 0x0000FF
  蓝色
- #define **WS2812_YELLOW** 0xFFFF00
  黄色
- #define **WS2812_PURPLE** 0x00FFFF
  紫色
- #define **WS2812_CYAN** 0xFF00FF
  青色
- #define **WS2812_ICEBLUE** 0xFF99FF
  雪蓝色

函数

- void WS2812_init (uint8_t NUM)
  初始化*WS2812*引脚并定义灯的数量
- void ws281x_delay (unsigned int delay_num)
  *ws281x*模块用到的延时函数
- void **ws281x_sendLow** (void)
  根据*WS281x*芯片时序图编写的发送*0*码，*1*码与*RESET*码的函数
- void **ws281x_sendHigh** (void)
  发送*1*码
- void **ws2811_Reset** (void)
  发送*RESET*码
- void ws281x_sendOne (uint32_t dat)
  发送点亮一个灯的数据（即*24bit*）
- void WS2812_SendColor (uint8_t NUM, uint8_t R, uint8_t G, uint8_t B)
  以*RGB*的形式发送颜色
- void WS2812_SendColor_u32 (uint8_t NUM, uint32_t Color)
  以*24*位*GRB*的形式发送颜色
- uint32_t WS2812_Wheel (uint32_t pos)
  色彩轮盘（*256*色）
- void WS2812_StreamColor (u8 pos)
  让所有灯珠显示以*pos*为起始的颜色流，在色轮上以步长*step*递增

5.11.1 详细描述

WS2812驱动头文件

作者

maker114

版本

0.1

日期

2025-06-16

5.11.2 函数说明

WS2812_init()

```
void WS2812_init (
            uint8_t NUM )
```

初始化WS2812引脚并定义灯的数量

参数

NUM	LED灯数量

WS2812_SendColor()

```
void WS2812_SendColor (
            uint8_t NUM,
            uint8_t R,
            uint8_t G,
            uint8_t B )
```

以RGB的形式发送颜色

参数

NUM	对应灯珠的编号，从0开始
R	R通道颜色值
G	G通道颜色值
B	B通道颜色值

WS2812_SendColor_u32()

```
void WS2812_SendColor_u32 (
            uint8_t NUM,
            uint32_t Color )
```

以24位GRB的形式发送颜色

参数

NUM	对应灯珠的编号，从0开始
Color	颜色的GRB编码（24位）

注解

常常搭配WS2812_Wheel函数使用

WS2812_StreamColor()

```
void WS2812_StreamColor (
            u8 pos )
```

让所有灯珠显示以pos为起始的颜色流，在色轮上以步长step递增

参数

pos	起始颜色

WS2812_Wheel()

```
uint32_t WS2812_Wheel (
            uint32_t pos )
```

色彩轮盘（256色）

参数

pos	颜色值

返回

uint32_t 输出的GRB编码，搭配WS2812_SendColor_u32函数使用

ws281x_delay()

```
void ws281x_delay (
            unsigned int delay_num )
```

ws281x模块用到的延时函数

参数

delay_num	:延时数 （示波器测量延时时间 = delay_num ∗ 440ns ）

ws281x_sendOne()

```
void ws281x_sendOne (
              uint32_t dat )
```

发送点亮一个灯的数据（即24bit）

参数

dat: 颜色的*24*位编码	

注解

不包含帧间延时

5.12 WS2812.h

浏览该文件的文档.

```
00001
00008 #ifndef __WS2812_H
00009 #define __WS2812_H
00010
00011 #include "delay.h"
00012 #include "usart.h"
00013 #include "sys.h"
00014
00015 void WS2812_init(uint8_t NUM);
00016
00017 void ws281x_delay(unsigned int delay_num);
00018
00019 void ws281x_sendLow(void);
00020
00021 void ws281x_sendHigh(void);
00022
00023 void ws2811_Reset(void);
00024
00025 void ws281x_sendOne(uint32_t dat);
00026
00027 void WS2812_SendColor(uint8_t NUM, uint8_t R, uint8_t G, uint8_t B);
00028
00029 void WS2812_SendColor_u32(uint8_t NUM, uint32_t Color);
00030
00031 uint32_t WS2812_Wheel(uint32_t pos);
00032
00033 void WS2812_StreamColor(u8 pos);
00034
00035 // 常用颜色GRB编码宏定义，搭配WS2812_SendColor_u32使用
00036 #define WS2812_BLACK 0x000000
00037 #define WS2812_WHITE 0xFFFFFF
00038 #define WS2812_RED 0x00FF00
00039 #define WS2812_GREEN 0xFF0000
00040 #define WS2812_BLUE 0x0000FF
00041 #define WS2812_YELLOW 0xFFFF00
00042 #define WS2812_PURPLE 0x00FFFF
00043 #define WS2812_CYAN 0xFF00FF
00044 #define WS2812_ICEBLUE 0xFF99FF
00045
00046 #endif
```

Index