7bot 机械臂伺服电机固件函数库

中文手册

Arm2.0 版本系列

张文超

7bot 利用 Serial 通信的舵机固件函数库

介绍

该固件函数库是一个函数包，它由宏，类，以及外部函数共同构成。该库中包含了每一个形参的讲解和一个应用示例。通过使用本固件函数库，用户无需掌握细节，也可以轻松对舵机进行操作，因此，该固件库可以大大减少用户的开发时间，从而降低开发成本。另外，对于学习需要的客户，使用该固件库，可以大大缩减入门时间，降低编程难度。

此库函数手册，一般分为三部分；

- 定义
- 函数库概述
- 函数库的应用描述

目录

1. 舵机应用层函数(steer.h)

1.0 舵机应用层 Steer 类中的变量

该类在 Steer.h 里定义，具体请查看该文件。

Table0．给出了 Steer 的公有变量列表

公有变量名(public)	描述
Position_Current[2]	获取当前位置的值存储在该变量中，Position_Current [0]中存储高位字节，单位 mm
Position_Target[2]	获取目标位置的值存储在该变量中，Position_Target [0]中存储高位字节，单位 mm
RunTime[2]	获取运行到目标位置的时间值存储在该变量中，RunTime [0]中存储高位字节，单位 ms，一般与目标位置一起写入，形成速度控制
Min_Angle_Limit[2]	最小角度限制，范围（0~4095）表示（0~360°），字节存储方式同上
Max_Angle_Limit[2]	最大角度限制，范围（0~4095）表示（0~360°），字节存储方式同上
Voltage	获取当前电压值，该功能暂未开放
Temperature	获取当前温度值，该功能暂未开放
Speed_Current[2]	获取当前速度值，该功能暂未开放
Electric_Current[2]	获取当前电流值，该功能暂未开放

Table1．给出了 Steer 的私有变量列表

私有变量名(pravite)	描述
id	舵机的 ID

1.1 Steer 类中的公有函数列表

Table2．给出了 Steer 的公有函数列表

函数名	描述
Steer	舵机通讯应用层构造函数：初始化 ID 和通讯串口
Steer_Ping	舵机应答函数，测试舵机的通信是否正常
Set_Steer_Max_Angle_Limit	设置舵机的最大角限制，建议设置在 4096 内，即 360°之内
Set_Steer_Min_Angle_Limit	设置舵机的最小角限制，建议设置在 4096，即 360°之内
Set_Steer_Torque_On	设置舵机舵机扭矩开关为开状态，即舵机获得扭矩
Set_Steer_Torque_Off	设置舵机舵机扭矩开关为关状态，即舵机失去扭矩
Set_Steer_position_runtime	舵机运行函数，配置该函数使舵机产生运动
Change_Steer_ID	改变舵机的 ID
Set_Steer_Reset	舵机恢复出厂设置
Get_Steer_All_Inf	得到舵机的所有当前信息

Get_Steer_Electric_Current_Inf	获得舵机的当前电流，获得的值存储在 Electric_Current[]数组里，具体的请看类定义里面的公有成员变量
Get_Steer_voltage_Inf	获得舵机的当前电压，获得的值存储在 voltage 里，具体的请看类定义里面的公有成员变量
Get_Steer_Temperature_Inf	获得舵机的当前温度，获得舵机的当前温度
Get_Steer_Position_Current_Inf	获得舵机的当前位置，获得的值存储在 Position_Current[]数组里，具体的请看类定义里面的公有成员变量
Get_Steer_Position_Target_Inf	获得舵机的目标位置，获得的值存储在 Position_Target[]数组里，具体的请看类定义里面的公有成员变量
Get_Steer_RunTime_Inf	获得舵机的运行时间，获得的值存储在 RunTime[]数组里，具体的请看类定义里面的公有成员变量
Get_Steer_Speed_Current_inf	获得舵机的当前速度，获得的值存储在 Speed_Current[]数组，具体的请看类定义里面的公有成员变量
Get_Steer_Angle_Limit_inf	获得舵机的最大/最小角度限制，获得的值存储在 Min_Angle_Limit[]和 Max_Angle_Limit[]两个数组里，具体的请看类定义里面的公有成员变量

1.1.0 函数 Steer

Table3. 描述了函数 Steer

Table3.

函数名	Steer
函数原型	Steer(byte ID=0 , HardwareSerial *serial = &Serial1)
功能描述	初始化 ID 和通讯串口
输入参数 1	id : 舵机的 ID 号
输入参数 2	Serial : 串口选择(需要根据自己的开发板实际确定)
返回值	无
先决条件	无
被调用函数	舵机底层通信函数 Set_Serial_init

例:

```
/**设置 id = 0，初始化串口 1,在创建该对象时运行该函数。**/
Steer    steer1( 0, &Serial1 );
```

1.1.1 函数 Steer_Ping

Table4.描述了函数 Steer_Ping

Table4.

函数名	Steer_Ping
函数原型	boolean Steer_Ping();
功能描述	舵机应答函数，测试舵机的通信是否正常
输入参数	无

返回值	返回布尔值：ture 为应答成功， false 为失败
先决条件	初始化一个 Steer 对象
被调用函数	舵机底层通信函数 ping

例:

```
/**设置 id = 0，初始化串口 1,同时检测该舵机是否通信（应答）正常**/
Steer    steer1( 0, &Serial1 );
steer1.Steer_Ping();
```

1.1.2 函数 Set_Steer_Max_Angle_Limit

Table5.描述了函数 Set_Steer_Max_Angle_Limit

Table5.

函数名	Set_Steer_Max_Angle_Limit
函数原型	void Set_Steer_Max_Angle_Limit(word max_angle);
功能描述	设置舵机的最大角限制
输入参数	max_angle: word 类型，设置舵机的最大值，范围（0~4095），即（0~360°）之内
返回值	无
先决条件	初始化一个 Steer 对象
被调用函数	舵机底层通信函数 write

例:

```
/**设置 id = 0，初始化串口 1,同时设置舵机最大偏转角度为 180°**/
Steer    steer1( 0, &Serial1 );
steer1. Set_Steer_Max_Angle_Limit (2047);
```

1.1.3 函数 Set_Steer_Min_Angle_Limit

Table6.描述了函数 Set_Steer_Min_Angle_Limit

Table6.

函数名	Set_Steer_Min_Angle_Limit
函数原型	void Set_Steer_Min_Angle_Limit(word max_angle);
功能描述	设置舵机的最小角限制
输入参数	min_angle: word 类型，设置舵机的最大值，范围（0~4095），即（0~360°）之内
返回值	无
先决条件	初始化一个 Steer 对象
被调用函数	舵机底层通信函数 write

例:

```
/**设置 id = 0，初始化串口 1,同时设置舵机最小偏转角度为 90°**/
Steer    steer1( 0, &Serial1 );
steer1. Set_Steer_Min_Angle_Limit (1024);
```

1.1.4 函数 Set_Steer_Torque_On

Table7.描述了函数 Set_Steer_Torque_On

Table7.

函数名	Set_Steer_Torque_On
函数原型	void Set_Steer_Torque_On();
功能描述	设置舵机舵机扭矩开关为开状态，即舵机获得扭矩
输入参数	无
返回值	无
先决条件	初始化一个 Steer 对象
被调用函数	舵机底层通信函数 write

例:

```
/**设置 id = 0，初始化串口 1,同时设置舵机的扭矩为开**/
Steer    steer1( 0, &Serial1 );
steer1. Set_Steer_Torque_On();
```

1.1.5 函数 Set_Steer_Torque_Off

Table8.描述了函数 Set_Steer_Torque_Off

Table8.

函数名	Set_Steer_Torque_Off
函数原型	void Set_Steer_Torque_Off();
功能描述	设置舵机舵机扭矩开关为开状态，即舵机失去扭矩
输入参数	无
返回值	无
先决条件	初始化一个 Steer 对象
被调用函数	舵机底层通信函数 write

例:

```
/**设置 id = 0，初始化串口 1,同时设置舵机的扭矩为关，即失去扭矩**/
Steer    steer1( 0, &Serial1 );
steer1. Set_Steer_Torque_Off();
```

1.1.6 函数 Set_Steer_position_runtime

Table9.描述了函数 Set_Steer_position_runtime

Table9.

函数名	Set_Steer_position_runtime
函数原型	void Set_Steer_position_runtime(word pos, word runtime)
功能描述	使舵机在 runtime 的时间内运行到目标位置 pos 处
输入参数 1	pos: word 类型，目标位置，范围（0~4095），即（0~360°）
输入参数 2	runtime: 运行时间，单位 ms,建议新手把该值调大，以防止运动过快伤人或损坏
返回值	无
先决条件	初始化一个 Steer 对象
被调用函数	舵机底层通信函数 sync_write

例:

```
/**设置 id = 0，初始化串口 1,同时设置舵机在 2000ms 内运行到 180°位置**/
```

```
Steer    steer1( 0, &Serial1 );
steer1. Set_Steer_position_runtime(2047, 2000);
```

1.1.7 函数 Change_Steer_ID

Table10.描述了函数 Change_Steer_ID

Table10.

函数名	Change_Steer_ID
函数原型	void Change_Steer_ID(byte new_id);
功能描述	改变舵机的 ID，注意：改变舵机的 ID 后，依然可以调用该对象控制该舵机，因为私有变量中的 id 同时被改变了
输入参数	new_id: 舵机的新 ID
返回值	无
先决条件	初始化一个 Steer 对象
被调用函数	舵机底层通信函数 write

例:

```
/**设置 id = 0，初始化串口 1,同时设置该舵机的新 id 为 2**/
Steer    steer1( 0, &Serial1 );
steer1. Change_Steer_ID (2);
```

1.1.8 函数 Set_Steer_Reset

Table11.描述了函数 Set_Steer_Reset

Table11.

函数名	Set_Steer_Reset
函数原型	void Set_Steer_Reset();
功能描述	舵机恢复出厂设置，复位后舵机的 ID 会被设为 0X01，注意：恢复出厂设置后，依然可以调用该对象控制该舵机，因为私有变量中的 id 同时被改变为 0x01
输入参数	无
返回值	无
先决条件	初始化一个 Steer 对象
被调用函数	舵机底层通信函数 reset

例:

```
/**设置 id = 0，初始化串口 1,同时使该舵机恢复出厂设置**/
Steer    steer1( 0, &Serial1 );
steer1. Set_Steer_Reset ();
```

1.1.9 函数 Get_Steer_Electric_Current_Inf（功能暂未开放）

Table12.描述了函数 Get_Steer_Electric_Current_Inf

Table12.

函数名	Get_Steer_Electric_Current_Inf
函数原型	void Get_Steer_Electric_Current_Inf ();
功能描述	获得舵机的当前电流，该功能暂未开放。如开放，获

	得的值存储在 Electric_Current[]数组里，具体的请看类定义里面的公有成员变量
输入参数	无
返回值	无
先决条件	初始化一个 Steer 对象
被调用函数	舵机底层通信函数 read

例:

```
/**设置 id = 0，初始化串口 1,同时获取该舵机的电流值**/
Steer    steer1( 0, &Serial1 );
steer1. Get_Steer_Electric_Current_Inf ();
```

1.1.10 函数 Get_Steer_voltage_Inf（功能暂未开放）

Table13.描述了函数 Get_Steer_voltage_Inf

Table13.

函数名	Get_Steer_voltage_Inf
函数原型	void Get_Steer_voltage_Inf ();
功能描述	获得舵机的当前电压，该功能暂未开放。如开放，获得的值存储在 voltage 里，具体的请看类定义里面的公有成员变量
输入参数	无
返回值	无
先决条件	初始化一个 Steer 对象
被调用函数	舵机底层通信函数 read

例:

```
/**设置 id = 0，初始化串口 1,同时获取该舵机的电压值**/
Steer    steer1( 0, &Serial1 );
steer1. Get_Steer_voltage_Inf ();
```

1.1.11 函数 Get_Steer_Temperature_Inf（功能暂未开发）

Table14.描述了函数 Get_Steer_Temperature_Inf

Table14.

函数名	Get_Steer_Temperature_Inf
函数原型	void Get_Steer_Temperature_Inf ();
功能描述	获得舵机的当前温度，该功能暂未开放。如开放，获得的值存储在 Temperature 里，具体的请看类定义里面的公有成员变量
输入参数	无
返回值	无
先决条件	初始化一个 Steer 对象
被调用函数	舵机底层通信函数 read

例:

```
/**设置 id = 0，初始化串口 1,同时获取该舵机的温度值**/
Steer    steer1( 0, &Serial1 );
```

steer1. Get_Steer_Temperature_Inf ();

1.1.12 函数 Get_Steer_Position_Current_Inf

Table15.描述了函数 Get_Steer_Position_Current_Inf

Table15.

函数名	Get_Steer_Position_Current_Inf
函数原型	void Get_Steer_Position_Current_Inf ();
功能描述	获得舵机的当前位置。如开放，获得的值存储在 Position_Current[]数组里，具体的请看类定义里面的公有成员变量
输入参数	无
返回值	无
先决条件	初始化一个 Steer 对象
被调用函数	舵机底层通信函数 read

例:

/**设置 id = 0，初始化串口 1,同时获取该舵机的当前位置**/

Steer steer1(0, &Serial1);

steer1. Get_Steer_Position_Current_Inf ();

1.1.13 函数 Get_Steer_Position_Target_Inf

Table16.描述了函数 Get_Steer_Position_Target_Inf

Table16.

函数名	Get_Steer_Position_Target_Inf
函数原型	void Get_Steer_Position_Target_Inf ();
功能描述	获得舵机的目标位置，获得的值存储在 Position_Target[]数组里，具体的请看类定义里面的公有成员变量
输入参数	无
返回值	无
先决条件	初始化一个 Steer 对象
被调用函数	舵机底层通信函数 read

例:

/**设置 id = 0，初始化串口 1,同时获取该舵机的目标位置**/

Steer steer1(0, &Serial1);

steer1. Get_Steer_Position_Target_Inf ();

1.1.14 函数 Get_Steer_RunTime_Inf

Table17.描述了函数 Get_Steer_RunTime_Inf

Table17.

函数名	Get_Steer_RunTime_Inf
函数原型	void Get_Steer_RunTime_Inf ();
功能描述	获得舵机运行到目标位置的运行时间，获得的值存储在 RunTime[]数组里，具体的请看类定义里面的公有

	成员变量
输入参数	无
返回值	无
先决条件	初始化一个 Steer 对象
被调用函数	舵机底层通信函数 read

例:

/**设置 id = 0，初始化串口 1,同时获取该舵机的运行到目标位置的运行时间**/

Steer steer1(0, &Serial1);

steer1. Get_Steer_RunTime_Inf ();

1.1.15 函数 Get_Steer_Speed_Current_inf（功能暂未开放）

Table18.描述了函数 Get_Steer_Speed_Current_inf

Table18.

函数名	Get_Steer_Speed_Current_inf
函数原型	void Get_Steer_Speed_Current_inf ();
功能描述	获得舵机的当前速度，该功能暂未开放。如开放，获得的值存储在 Speed_Current []数组里，具体的请看类定义里面的公有成员变量
输入参数	无
返回值	无
先决条件	初始化一个 Steer 对象
被调用函数	舵机底层通信函数 read

例:

/**设置 id = 0，初始化串口 1,同时获取该舵机的当前速度**/

Steer steer1(0, &Serial1);

steer1. Get_Steer_Speed_Current_inf ();

1.1.16 函数 Get_Steer_Angle_Limit_inf

Table19.描述了函数 Get_Steer_Angle_Limit_inf

Table19.

函数名	Get_Steer_Angle_Limit_inf
函数原型	void Get_Steer_Angle_Limit_inf ();
功能描述	获得舵机的最大最小角度限制，获得的值存储在 Min_Angle_Limit[]和 Max_Angle_Limit[]两个数组里，具体的请看类定义里面的公有成员变量
输入参数	无
返回值	无
先决条件	初始化一个 Steer 对象
被调用函数	舵机底层通信函数 read

例:

/**设置 id = 0，初始化串口 1,同时获取该舵机的最大最小角度限制**/

Steer steer1(0, &Serial1);

steer1. Get_Steer_Angle_Limit_inf ();

2.舵机底层通信协议函数(Steer_protocol.h)

如果你想了解舵机通信的底层细节，请继续往下阅读，如果你仅仅是想用好舵机，那么参考第一章应用层的内容就可以解决大部分问题，如果你有特殊原因必须了解舵机控制细节，请最好配合资料中的《7bot 机械臂舵机通信协议》，以及源程序 Steer_protocol.c 文件和 Steer_protocol.h 文件来阅读，源文件中给出了较为完整的注释，你可以看到我们对于底层寄存器的操作，本文档仅仅是为了方便查阅舵机函数以及基本用法。

2.0 舵机底层通信协议中的指令和寄存器的宏定义

Table20．给出了 Steer_protocol.h 的全部指令描述

指令	宏定义	描述
0X01	INSTRUCTION_PING	查询指令
0X02	INSTRUCTION_READ_DATA	读指令
0X03	INSTRUCTION_WRITE_DATA	写指令
0X04	INSTRUCTION_REG_WRITE	异步写指令
0X05	INSTRUCTION_ACTION	触发异步写指令
0X06	INSTRUCTION_RESET	复位指令
0X83	INSTRUCTION_SYNC_WRITE	同步写指令

Table21．给出了 Steer_protocol.h 的寄存器地址描述

寄存器地址	宏定义	描述
0X05	ID_REG	ID 的地址：通过修改该位置的值，可以修改舵机的地址（地址范围:: 0 ~253
0X09	MIN_ANGLE_LIMIT_H	最小角度限制的高字节存储位置
0X0A	MIN_ANGLE_LIMIT_L	最小角度限制的低字节存储位置
0X0B	MAX_ANGLE_LIMIT_H	最大角度限制的高字节存储位置
0X0C	MAX_ANGLE_LIMIT_L	最大角度限制的低字节存储位置
0X10	MAX_TORQUE_H	最大扭矩高字节存储位置
0X11	MAX_TORQUE_L	最大扭矩低字节存储位置
0X12	DEFAULT_SPEED	调整速度位置
0X14	MIDDLE_POSITION_H	中位调整高字节存储位置
0X15	MIDDLE_POSITION_L	中位调整低字节存储位置
0X28	TORQUE_SWITCH	扭矩开关（1 开，0 关）
0X2A	TARGET_POSITION_H	目标位置高字节存储位置
0X2B	TARGET_POSITION_L	目标位置低字节存储位置
0X38	CURRENT_POSITION_H	当前位置高字节存储位置
0X39	CURRENT_POSITION_L	当前位置低字节存储位置
0X41	TARGET_SPEED	速度调整

0XFE	BROADCAST_ADDR	广播地址

2.1 舵机底层通信协议层 Steer_protocol 类中的变量

该类在 Steer_protocol.h 里定义，具体请查看该文件。

Table22．给出了 Steer_protocol 的公有变量列表

公有变量名(public)	描述
无	无

Table23．给出了 Steer_protocol 的私有变量列表

私有变量名(pravite)	描述
svSer	串口选择：例如 &Serial1

2.2 Steer_protocol 类中的公有函数列表

Table24．给出了 Steer 的公有函数列表

函数名	描述
Steer_protocol	舵机通讯协议构造函数：初始化通讯串口
Set_Serial_init	初始化通讯串口
begin	与电脑通信初始化，用户仅在测试舵机的时候使用
Check_Sum	校验和函数
ping	工作状态查询函数
read	读取舵机状态函数
reset	舵机复位函数
write	写函数
sync_write	同步写函数（广播）

2.2.0 函数 Steer_protocol() = default;（注意:该函数重载，下个表格介绍其重载函数）

Table25.描述了函数 Steer_protocol

Table25.

函数名	Steer_protocol
函数原型	Steer_protocol() = default;
功能描述	默认构造函数
输入参数	无
返回值	无
先决条件	初始化一个 Steer_protocol 对象
被调用函数	无

例:

```
/**创建一个 Steer_protocol 对象并默认初始化**/
Steer_protocol    steer_protocol();
```

2.2.1 函数 Steer_protocol;（注意:该函数重载，下个表格介绍其重载函数）

Table26.描述了函数 Steer_protocol

Table26.

函数名	Steer_protocol
函数原型	Steer_protocol(HardwareSerial *serial, long timeout)
功能描述	舵机通讯协议构造函数：初始化通讯串口
输入参数 1	Serial: 串口选定
输入参数 2	Timeout：串口超时
返回值	无
先决条件	初始化一个 Steer_protocol 对象
被调用函数	无

例:

/**创建一个 Steer_protocol 对象并设置其通信串口为 serial1,且超时时间设为 10 毫秒**/

Steer_protocol steer_protocol(&Serial1, 10);

2.2.2 函数 Set_Serial_init

Table27.描述了函数 Set_Serial_init

Table27.

函数名	Set_Serial_init
函数原型	void Set_Serial_init(HardwareSerial *serial)
功能描述	为了给其父类 Steer 提供一个初始化接口
输入参数	Serial: 串口选定
返回值	无
先决条件	初始化一个 Steer_protocol 对象
被调用函数	无

例:

/**创建一个 Steer_protocol 对象并初始化它**/

Steer_protocol steer_protocol;

steer_protocol. Set_Serial_init(&Serial1);

2.2.2 函数 begin

Table28.描述了函数 begin

Table28.

函数名	begin
函数原型	void begin(HardwareSerial *serial, long timeout)
功能描述	用户仅在测试舵机的时候使用,和电脑通信
输入参数 1	Serial: 串口选定
输入参数 2	Timeout：串口超时
返回值	无
先决条件	初始化一个 Steer_protocol 对象
被调用函数	无

例:

/**创建一个 Steer_protocol 对象并设置其通信串口为 serial1,且超时时间设为 10 毫秒，同时初始化板子与电脑的通信**/

Steer_protocol steer_protocol(&Serial1, 10);

steer_protocol,begin(&Serial1, 10);

2.2.3 函数 Check_Sum

说明：校验和公式：Check Sum = ~ (ID + Length + Instruction + Parameter1 + ... Parameter N);具体的请查看《7bot 机械臂舵机通信协议》

Table29.描述了函数 Check_Sum

Table29.

函数名	Check_Sum
函数原型	byte Check_Sum(byte *buf, byte len);
功能描述	求校验和
输入参数 1	buf: 需要校验的数组地址
输入参数 2	len： 数组长度
返回值	数据类型 byte：校验和
先决条件	初始化一个 Steer_protocol 对象
被调用函数	无

例:

```
/**创建一个 Steer_protocol 对象并输出数据 buf 的最后一个数字为校验和 ， 设 byte buf[] = {1,2,3,4,5,6} 此时 len = 5 **/
Steer_protocol steer_protocol(&Serial1, 10);
steer_protocol.Check_Sum (buf, 5);
```

2.2.4 函数 ping

Table30.描述了函数 ping

Table30.

函数名	ping
函数原型	boolean ping(byte id, byte *data)
功能描述	工作状态查询函数，测试舵机是否应答
输入参数 1	id: 舵机的 ID
输入参数 2	data：输入类型为 byte 的指针，获得舵机工作状态
返回值	true: 舵机有应答，通信正常 false: 舵机无应答，通信不正常
先决条件	初始化一个 Steer_protocol 对象
被调用函数	无

例:

```
/**创建一个 Steer_protocol 对象并设 byte *dat; 查询 ID 为 1 的舵机 的工作状态**/
Steer_protocol steer_protocol(&Serial1, 10);
steer_protocol. ping(0x01 , dat);
```

2.2.5 函数 read

Table31.描述了函数 read

Table31.

函数名	read
函数原型	Boolean read(byte id, byte regStartAddr, byte *data, byte readlen)

功能描述	读取舵机状态函数
输入参数 1	id: 舵机的 ID
输入参数 2	regStartAddr：需读取信息的内存开始地址
输入参数 3	data：输入类型为 byte 的指针，存储读取的信息
输入参数 4	readlen：读取的数据长度
返回值	true: 读取成功 false: 读取失败
先决条件	初始化一个 Steer_protocol 对象
被调用函数	无

例:

/**创建一个 Steer_protocol 对象并设舵机 ID = 1；在控制表里从地址 0X38 处读取二个字节,Steer_protocol Steer1; byte dat[2]; **/

```
Steer_protocol  steer_protocol(&Serial1, 10);
steer_protocol. read(0x01 , 0x38 , dat , 0x02);
```

2.2.6 函数 reset

Table32.描述了函数 reset

Table32.

函数名	reset
函数原型	void reset(byte id)
功能描述	舵机复位函数:使舵机恢复出厂设置
输入参数 1	id: 舵机的 ID
返回值	Void
先决条件	初始化一个 Steer_protocol 对象
被调用函数	write

例:

/**创建一个 Steer_protocol 对象并设舵机 ID = 1 恢复出厂设置**/

```
Steer_protocol  steer_protocol(&Serial1, 10);
steer_protocol. reset (0x01);
```

2.2.7 函数 write

Table33.描述了函数 write

Table33.

函数名	write
函数原型	void write(byte id, byte regStartAddr, byte *buf, byte bufLen)
功能描述	写函数
输入参数 1	id: 舵机的 ID
输入参数 2	regStartAddr：需写入信息的内存开始地址
输入参数 3	buf:输入类型为 byte 的数组指针，存储需要写的信息
输入参数 4	bufLen:写入数据的长度
返回值	void
先决条件	初始化一个 Steer_protocol 对象

被调用函数	无

例:

/**创建一个 Steer_protocol 对象并设舵机 ID = 1；在控制表里从地址 0X2A 处写入二个字节，为目标位置, byte dat[2] = { 0x00 , 0xff }**/

Steer_protocol steer_protocol(&Serial1, 10);

steer_protocol. write(0x01 , 0X2A , dat , 0x02);

2.2.8 函数 sync_write

Table34.描述了函数 sync _write

Table34.

函数名	sync_write
函数原型	void sync_write(byte regStartAddr, byte *buf, byte svNum, byte perDataLen)
功能描述	同步写函数：可以给一个或多个舵机同时写入几个数据，常用的是同时写入目标位置和运行时间形成速度控制
输入参数 1	regStartAddr：需写入信息的内存开始地址
输入参数 2	buf:输入类型为 byte 的数组指针，存储需要写的信息
输入参数 3	svNum：需要同步写入的舵机数量
输入参数 4	perDataLen:每个舵机需要写入数据的长度
返回值	void
先决条件	初始化一个 Steer_protocol 对象
被调用函数	无

例:

/**创建一个 Steer_protocol 对象并设在控制表里从地址 0X2A 处写入二个字节，为目标位置, byte dat[] = { 0x01,0x00 , 0xff.....0x06,0x00,0xff }; **/

Steer_protocol steer_protocol(&Serial1, 10);

steer_protocol. sync_write(0X2A , buf , 6 , 3);