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

[7bot机械臂伺服电机固件函数库 1](#_Toc492321056)

[基于利用Serial通信的新型舵机固件函数库 2](#_Toc492321060)

[1.舵机应用层函数(steer.h) 4](#_Toc492321061)

[1.0舵机应用层Steer类中的变量 4](#_Toc492321062)

[1.1 Steer类中的公有函数列表 4](#_Toc492321063)

[1.1.0函数Steer 5](#_Toc492321064)

[1.1.1函数Steer\_Ping 5](#_Toc492321065)

[1.1.2函数Set\_Steer\_Max\_Angle\_Limit 6](#_Toc492321066)

[1.1.3函数Set\_Steer\_Min\_Angle\_Limit 6](#_Toc492321067)

[1.1.4函数Set\_Steer\_Torque\_On 6](#_Toc492321068)

[1.1.5函数Set\_Steer\_Torque\_Off 7](#_Toc492321069)

[1.1.6函数Set\_Steer\_position\_runtime 7](#_Toc492321070)

[1.1.7函数Change\_Steer\_ID 8](#_Toc492321071)

[1.1.8函数Set\_Steer\_Reset 8](#_Toc492321072)

[1.1.9函数Get\_Steer\_Electric\_Current\_Inf（功能暂未开放） 8](#_Toc492321073)

[1.1.10函数Get\_Steer\_voltage\_Inf（功能暂未开放） 9](#_Toc492321074)

[1.1.11函数Get\_Steer\_Temperature\_Inf（功能暂未开发） 9](#_Toc492321075)

[1.1.12函数Get\_Steer\_Position\_Current\_Inf 10](#_Toc492321076)

[1.1.13 函数Get\_Steer\_Position\_Target\_Inf 10](#_Toc492321077)

[1.1.14函数Get\_Steer\_RunTime\_Inf 10](#_Toc492321078)

[1.1.15函数Get\_Steer\_Speed\_Current\_inf（功能暂未开放） 11](#_Toc492321079)

[1.1.16函数Get\_Steer\_Angle\_Limit\_inf 11](#_Toc492321080)

[2.舵机底层通信协议函数(Steer\_protocol.h) 12](#_Toc492321081)

[2.0舵机底层通信协议中的指令和寄存器的宏定义 12](#_Toc492321082)

[2.1舵机底层通信协议层Steer\_protocol类中的变量 13](#_Toc492321083)

[2.2 Steer\_protocol类中的公有函数列表 13](#_Toc492321084)

[2.2.0函数Steer\_protocol() = default 13](#_Toc492321085)

[2.2.1函数Steer\_protocol 13](#_Toc492321086)

[2.2.2函数Set\_Serial\_init 14](#_Toc492321087)

[2.2.2函数begin 14](#_Toc492321088)

[2.2.3函数Check\_Sum 15](#_Toc492321089)

[2.2.4函数ping 15](#_Toc492321090)

[2.2.5函数read 15](#_Toc492321091)

[2.2.6函数reset 16](#_Toc492321092)

[2.2.7函数write 16](#_Toc492321093)

[2.2.8函数sync\_write 17](#_Toc492321094)

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. 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. 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. 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. 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. 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. 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. 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. 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. 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. 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. 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. 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. 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. 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. 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. 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. 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);