**7bot机械臂舵机篇应用层库函数**

**中文教程**

Arm2.0版本系列

张文超

**机械臂舵机篇实战教程**

本实战教程主要采用舵机应用层函数实现，具体函数使用请参考《舵机库函数中文手册》。

为了更好更方便的测试舵机，建议大家单独买一块开发板和一个舵机以用来测试，当然，你也可以用机械臂的舵机来测试，如果你要使用机械臂上的舵机来测试，建议你刚开始使用ID为5的舵机，也就是安装连接末端机构的那个舵机（末端机构上的舵机ID为6），因为这个舵机可以360度旋转而不容易损坏。

注意事项：

1.请先用电源线对机械臂和舵机进行上电，再接USB线。防止机械臂需求电流过大，烧毁您的电脑USB端口。

目录

[准备工作 4](#_Toc492842811)

[第一章 舵机应答测试（ping） 9](#_Toc492842812)

[1.1 舵机测试应答过程简介 10](#_Toc492842813)

[1.2 用测试应答为例子去进行中文文档查阅教学 10](#_Toc492842814)

[1.3 Arduino上编写程序 12](#_Toc492842815)

[1.4 下载验证 12](#_Toc492842816)

[第二章 改变舵机ID实验（change\_ID） 15](#_Toc492842817)

[2.1 改变舵机ID简介 16](#_Toc492842818)

[2.2 改变舵机ID函数介绍 16](#_Toc492842819)

[2.3 Arduino例程文件编写 18](#_Toc492842820)

[2.4 下载验证 18](#_Toc492842821)

[第三章 舵机位置控制实验（Set\_Steer\_position\_runtime） 21](#_Toc492842822)

[3.1 控制舵机简介 22](#_Toc492842824)

[3.2 控制舵机函数介绍 22](#_Toc492842825)

[3.3 Arduino例程文件编写 23](#_Toc492842826)

[3.4 下载验证 23](#_Toc492842827)

[第四章 舵机状态信息获取实验（Get\_Steer\_XXX\_Inf） 26](#_Toc492842828)

[4.1 舵机状态信息获取实验简介 27](#_Toc492842830)

[4.2 舵机状态信息获取函数介绍 27](#_Toc492842831)

[4.3 Arduino例程文件编写 29](#_Toc492842832)

[4.4 下载验证 30](#_Toc492842833)

[第五章 舵机恢复出厂设置实验（Set\_Steer\_Reset） 32](#_Toc492842834)

[5.1 舵机恢复出厂设置实验简介 33](#_Toc492842836)

[5.2 舵机恢复出厂设置函数介绍 33](#_Toc492842837)

[5.3 Arduino例程文件编写 33](#_Toc492842838)

[5.4 下载验证 34](#_Toc492842839)

# 准备工作

准备工作是把舵机的函数库和示例放到arduino的library下面，如下图步骤所示：

第一步，在我们的资源中找到下图中的7bot程序源码文件夹，双击打开。

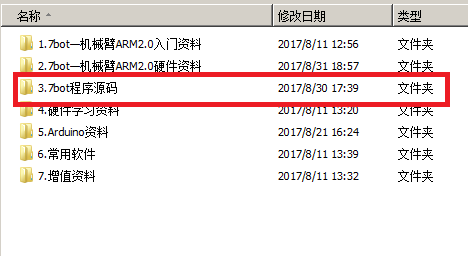


图0.1

第二步 找到如下所示文件夹，双击打开。

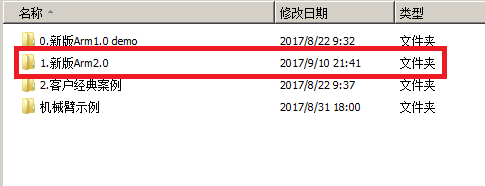


图0.2

第三步 找到如下所示文件夹，复制。

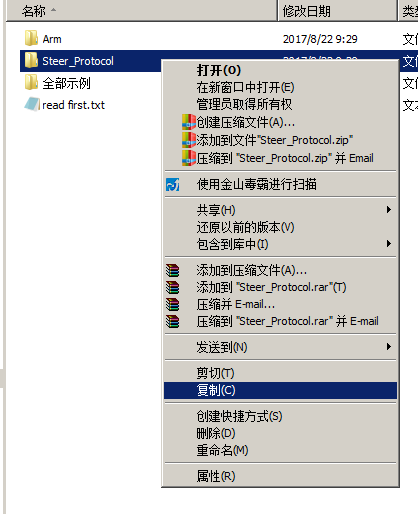


图0.3

第四步 在我们的arduino安装目录下找到如下所示文件夹，双击打开。

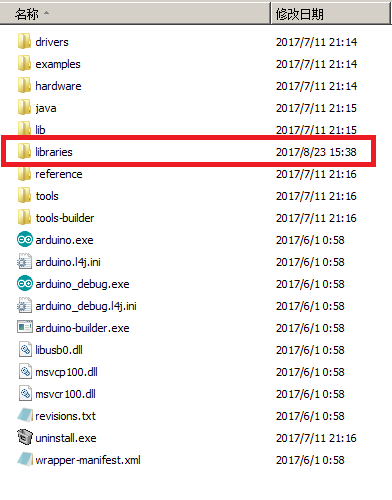


图0.4

第五步 将我们刚才复制的文件夹粘贴到该文件夹下，如下图所示：

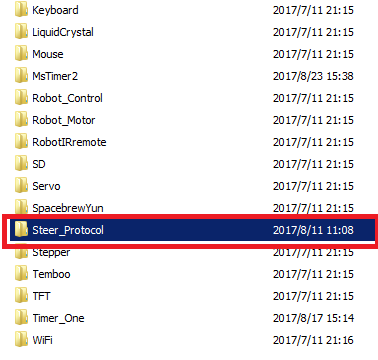


图0.5

第六步 关闭所有的arduino窗口，重新打开，我们在我们的示例下找到我们刚才复制的文件名，看到我们的示例展示，如下图所示。



图0.6

准备工作完成。

# 舵机应答测试（ping）

通过一个最简单的舵机函数，带领大家一起去响应我们的第一个舵机，开启我们学习机械臂的大门，下面我们开始迈入机械臂大门的第一步，舵机应答测试。

* 1. 舵机测试应答过程简介
  2. 用测试应答为例子去进行中文文档查阅教学
  3. Arduino上编写程序
  4. 下载验证

## 1.1 舵机测试应答过程简介

舵机应答测试是为了测试舵机是否通信正常（一般也用来判断舵机是否正常），所谓的舵机应答测试，也就是我们发送给舵机一组数据，这组数据包括舵机的ID，那么相应ID的舵机就会返回给我们一组数据，通过校验这组数据，我们就可以判断这个舵机是否应答正常，而这个过程我们的库函数已经帮我们封装完毕，我们只需要调用库函数，就可以判断我们的舵机是否通信正常。

## 1.2 用测试应答为例子去进行中文文档查阅教学

首先，我们通过查阅《7bot机械臂伺服电机固件函数库》的舵机应用层函数的公有函数列表。如下所示，找到该舵机的相应函数Steer\_Ping

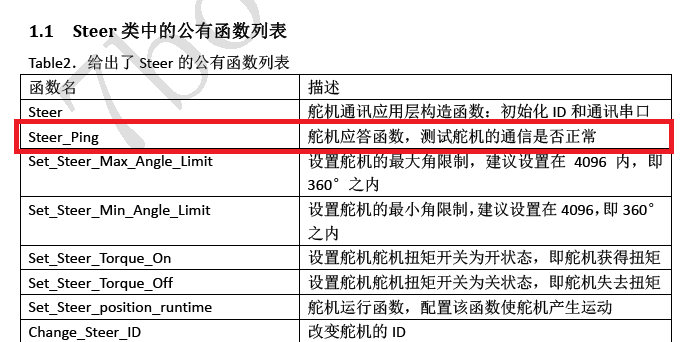


图1.2.1

其次，我们返回目录，在目录中找到相应函数

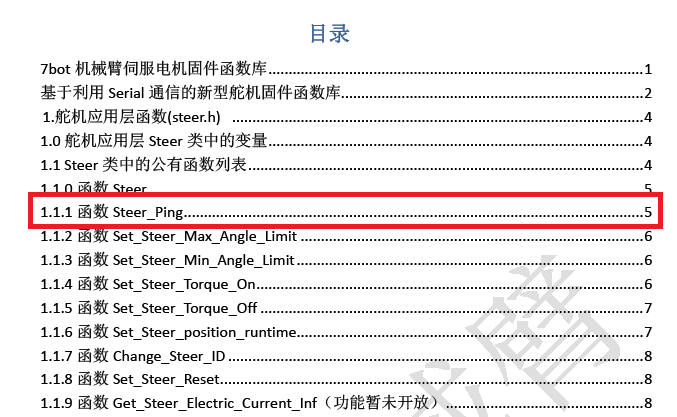


图1.2.2

然后点击，就会直接跳转到这个函数的详细使用说明的地方，如下所示



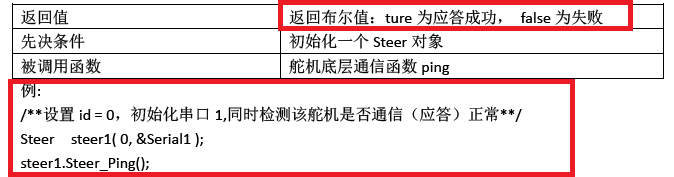


图1.2.3

从上表中我们可以看到这个函数的函数原型，我们知道没有输入参数，返回值是boolean类型，并且它有一个先决条件，就是传建一个Steer对象，我们参考中文手册中的例程，来在arduino上写出我们的工程代码。

## 1.3 Arduino上编写程序

因为我们的先决条件是初始化一个舵机对象，我们用相同的办法去查阅Steer的默认构造函数，从而去初始化一个Steer对象。

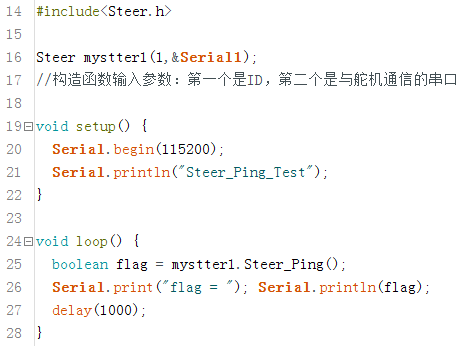


图1.3.1

我们首先初始化一个Steer舵机对象，构造函数如上图第17行注释所示，为了看到应答结果，我们初始化了串口Serial用于显示结果。

可以看到我们在loop中直接调用Steer\_Ping()方法去测试应答。

注：具体例程代码请参考例程文件中的steer\_ping\_test.ino

重要 注：关于对舵机函数这样封装的解释，我们采用把ID属性放到私有，是因为一般情况下，我们不会改变舵机的ID值，这样的话，我们用舵机类去初始化对象时，我们就不必要去管它的ID，我们知道我们创建的对象对应于现实的那个实物舵机，我们就可以全心的投入到舵机的控制中去，同时带来的弊端就是关于ID操作的灵活性没那么好了，我们在下章会看到我们去修改ID。

## 1.4 下载验证

如下图所示找到该程序并打开：



图1.4.0

下载程序后打开串口显示器，我们可以看到如下结果：

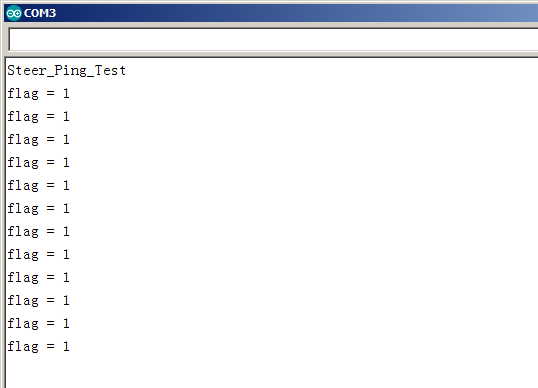


图1.4.1

每一秒去访问一次舵机，舵机给出一次应答。

flag = 1，表示舵机响应成功。

# 改变舵机ID实验（change\_ID）

第二个依旧是简单基础实验，我们去利用改变舵机ID的函数去改变舵机的ID，该函数我们也已经封装在函数库里，请按照第一章的步骤去查看中文手册，在这里就不一一赘述。

2.1 改变舵机ID简介

2.2 改变舵机ID函数介绍

2.3 Arduino例程文件编写

2.4 下载验证

## 2.1 改变舵机ID简介

改变舵机ID是把原有的舵机ID改变为我们想要的ID，通过调用Change\_Steer\_ID函数来改变ID。

注意：改变舵机ID后我们依然可以用该对象去控制该舵机，因为该对象的私有变量中的id被同时改变了。

## 2.2 改变舵机ID函数介绍

通过查阅手册，我们看到如下函数：

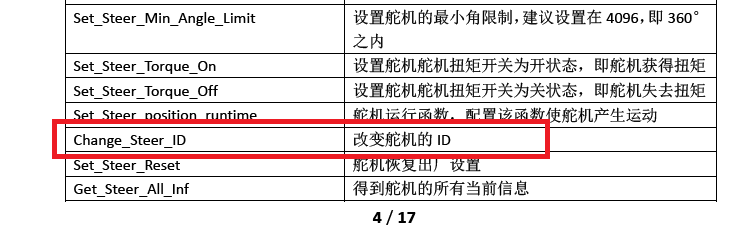


图2.2.0

具体查阅的过程参考第一章的内容，我们查阅到如下函数原型：

void Change\_Steer\_ID( byte new\_id);

对于这个函数，其实没什么可讲的，我们只要把舵机的新ID输入，它的ID就改变了，那么我们再来看一下这个函数的内部是什么样子的。

void Steer::Change\_Steer\_ID( byte new\_id){

byte new\_id\_ = new\_id;

write(id, 0x05, &new\_id\_, sizeof(new\_id\_));

id = new\_id;

}

其中的id变量是Steer的私有变量，其实我们在对舵机做任何操作时，即我们调用几乎所有的应用层函数时，里面都会调用id，我们为了id不被意外改变，所以我们把id设为私有属性，在初始化Steer的时候，由构造函数进行对id的赋值操作。

再次强调，改变舵机的ID依然可以用该对象去调用该舵机，因为函数中有这么一条语句（id = new\_id;），请看刚才给出的函数最后一行，就是说我们在改变舵机的ID时，我们同时改变了舵机对象内部的私有变量id的值。

关于这个函数怎么使用，就是初始化一个舵机对象，然后用该对象去调用这个函数，例如：

Steer str1(1,&Serial1); //我们初始化一个舵机，ID为1

Str. Change\_Steer\_ID(2); //我们将它的舵机ID改为2

注意，我们一般初始化的时候，一般舵机ID为多少，我们就初始化它的str后面的数字是多少，比如；舵机ID为1，我们对象就命名为str1，这样我们一般看到舵机对象后面的数字，我们就可以知道舵机的id（因为我们在正常使用的时候不会改变舵机的ID，所以我们这样命名，这一节课是改变舵机ID，所以我们可以根据自己的想法随便命名）。

## 2.3 Arduino例程文件编写



图2.3.0

如果图不清晰，请参考例程steer\_change\_id\_test.ino

例程上面的注释很完整，就不需要再多写什么。

## 2.4 下载验证

在示例中找到该函数，如下图所示：



图2.4.0

验证结果：

根据提示输入新的ID号，如下图，我输入16。

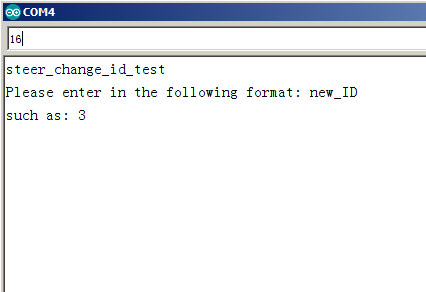


图2.4.1

按下回车，得到下面结果代表改变ID下成功。

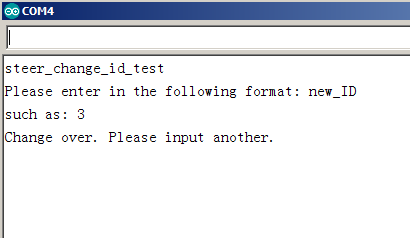


图2.4.2

我们可以继续输入，继续改变。

# 舵机位置控制实验

# （Set\_Steer\_position\_runtime）

在这一章，我们就要开始去控制舵机了，我们只需要调动上面写的那个函数，你就可以轻松控制舵机了，下面让我们一起开始吧

3.1 控制舵机简介

3.2 控制舵机函数介绍

3.3 Arduino例程文件编写

3.4 下载验证

## 3.1 控制舵机简介

控制舵机：

控制什么？控制舵机的旋转位置

怎么控制？直接输入位置的信息

有问题吗？有，那输入了位置，它多长时间运行到该位置，可以控制吗？当然可以，所以，我们想要控制舵机，就需要给舵机输入两个信息，即：舵机的位置和时间信息。

## 3.2 控制舵机函数介绍

那么我们通过简介，我们知道需要控制一个舵机应该至少有位置和时间信息，那么我们查阅舵机的中文手册，我们看到该函数，查阅过程参照第一章，我直接给出该函数的原型。

void Set\_Steer\_position\_runtime(word pos, word runtime);

确实只有两个输入参数，第一个参数pos是位置信息，而参数runtime是运行时间，我们来详细介绍下这两个参数。

pos：位置参数，值范围（0~4096），对应角度范围（0~360°）

runtime：运行时间，值范围（0~30000），单位ms

使用方法：我们初始化一个舵机对象，直接调用该函数，输入相应的位置和时间信息，就可以控制舵机转到相应位置。如下所示：

Steer str5(5, &Serial1); //初始化一个ID为5的舵机

str5. Set\_Steer\_position\_runtime(2047, 2000); //两秒内运行到中位

## 3.3 Arduino例程文件编写

打开例程steer\_move\_to\_position\_test，我们看到如下程序：

图3.3.0

上图注释清楚，在此不再赘述。

## 3.4 下载验证

如下找到该程序示例，点击打开并下载到芯片中：

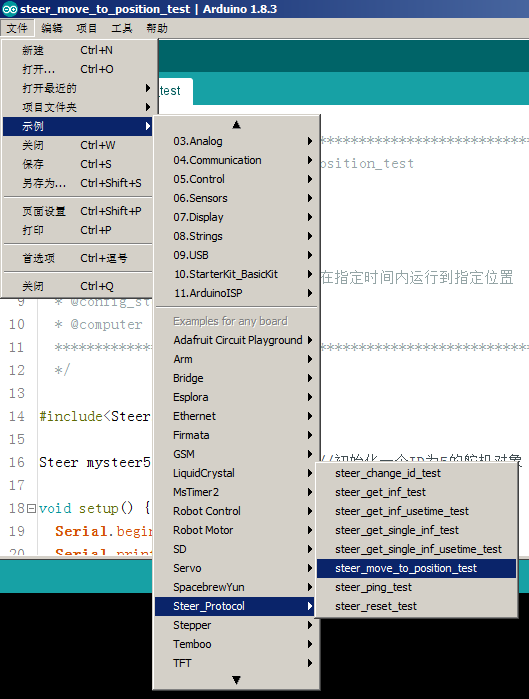


图3.4.0

烧写完成后打开串口显示器，根据提示的格式输入相关值，如下图所示：

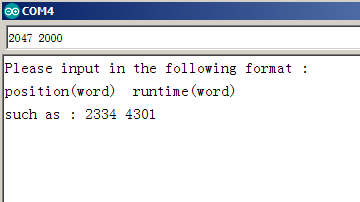


图3.4.1

按下回车键，听到ID为5的舵机吱吱吱的开始转了就对了。

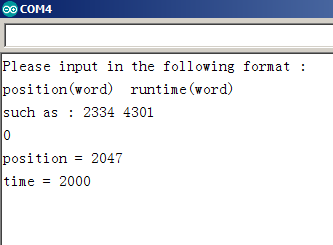


图3.4.2

如果你没有看到你的舵机转动，一般有3种可能：第一种，舵机本来就处于这个位置，那么舵机就不会转了。第二种，舵机的ID不对，你搞错舵机的ID号了，建议你用第一章的例程测下就知道了。第三种，舵机坏了，这种情况很少，但也有可能。

# 舵机状态信息获取实验

# （Get\_Steer\_XXX\_Inf）

在这一章，我们将学习获取舵机的状态信息，请注意看简介。

4.1 舵机状态信息获取实验简介

4.2 舵机状态信息获取函数介绍

4.3 Arduino例程文件编写

4.4 下载验证

## 4.1 舵机状态信息获取实验简介

获取舵机的状态信息：

啥叫状态信息？舵机的当前位置，目标位置，温度，角度限制等等都叫舵机的状态信息。

如何去获取舵机的状态信息？我们在库函数中已经封装好了舵机信息获取函数，调用舵机的信息获取函数，就可以得到相关舵机信息。

获取到的信息存储在哪里？获得的舵机信息存储在Steer类的公有变量里，我们可以去查看舵机的中文手册就清楚了。我们通过调用该公有变量，就可以去获取相关信息了。

## 4.2 舵机状态信息获取函数介绍

这里不得不祭出我们的大招了，我们查阅手册，看下图：

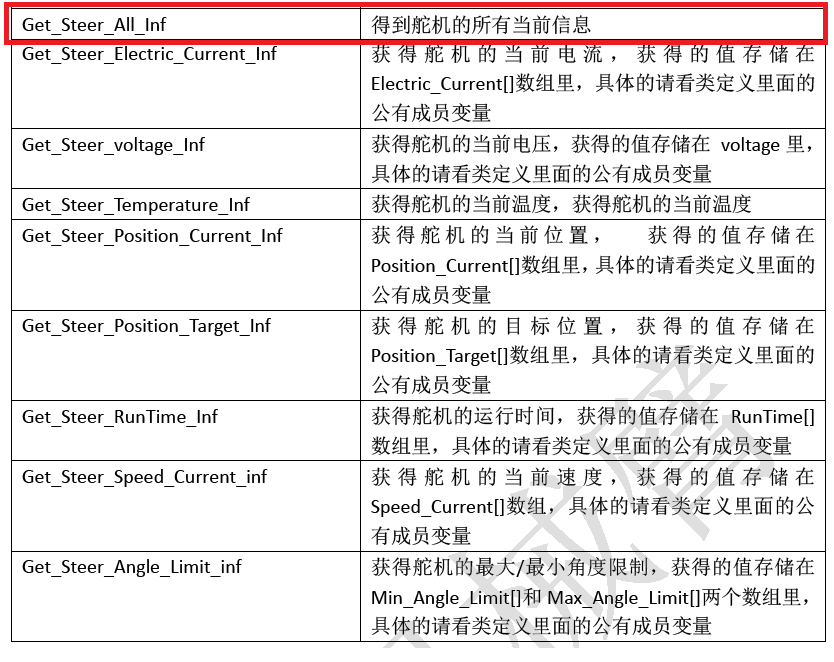


图4.2.0

看上图，它们全是获取舵机信息函数，注意看红色标注的函数，调用它一个函数就可以获取到全部信息，即，相当于把红色注释函数下边的所有函数挨个调用一遍。

而红色框框下边的函数，可以单独获取某一种信息。

为什么我们有了可以获取全部数据的函数，又要提供分个信息获取的函数呢？

因为时间，对，获取所有的信息调用Get\_Steer\_All\_Inf函数要耗时13ms左右，这对于有些对实时性要求比较高的项目来说是不可忍受的，所以我们又开放了下面的函数，因为单个调用下面的函数耗时大约在1.3ms附近。

我们通过查阅中文手册就可以知道它们的原型和使用方法，它们的原型都是这样的，如下所示：

void Get\_Steer\_XXX\_Inf();

我们以获取舵机当前位置信息为例如下所示；

Steer str5(5, &Serial1); //初始化一个ID为5的舵机对象

str5. Get\_Steer\_Position\_Current\_Inf (); //得到舵机的当前位置信息

我们学会用舵机的方法去获取函数信息，那么我们接下来看看这些数据放到了哪里，我们该如何去使用它。如下图所示：

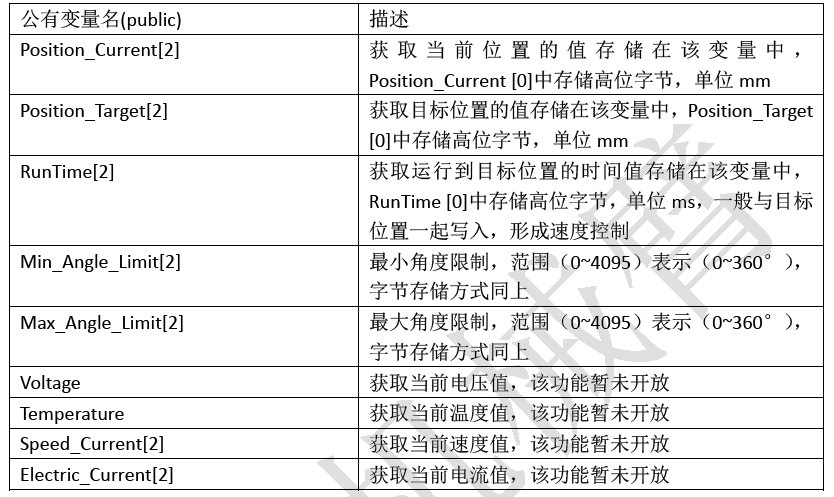


图4.2.1

上图展示了Steer类的公有变量，也就是存储舵机状态信息的变量。如果我们调用图4.2.0中的红色框框中的函数，也就是Get\_Steer\_All\_Inf函数，上面的所有变量值都会更新到当前状态值。如果我们只是调用获取某种状态信息的函数，那么只有对应的变量值会更新到当前状态值。

我在此在以当前位置为例，输出该舵机的当前位置信息。

假设我们以Serial串口和电脑的串口通信，那么我们将输出舵机的当前位置信息，如下所示：

Steer str5(5, &Serial1); //初始化一个ID为5的舵机对象

str5. Get\_Steer\_Position\_Current\_Inf (); //得到舵机的当前位置信息

Serial.println((((word) Position\_Current[0])<<8 ) + Position\_Current[1]);

//输出舵机的当前位置信息

## 4.3 Arduino例程文件编写

打开舵机应用层例程steer\_get\_inf\_test程序如下所示：



图4.3.0

关于显示我们写在了函数display\_all\_inf();里，因为该函数横向有点长，就不在这里贴图了，请到该例程中查看。

## 4.4 下载验证

如下图所示找到并打开该示例：

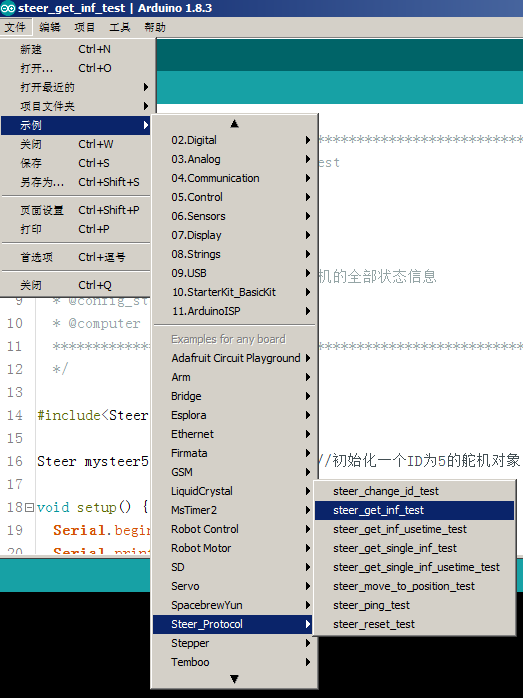


图4.4.0

烧写完并打开串口显示器，如下图所示，我们输入任意值，就会更新一次全部状态值：

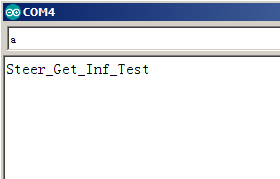


图4.4.1

按下回车键，读取舵机现在的状态信息，如下如所示：

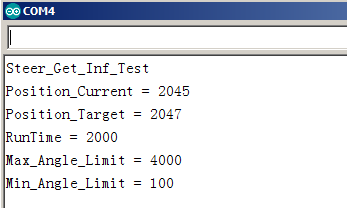


图4.4.2

上图中我们可以看到目标位置为2047，但是当前位置为2045，这是正常显现，舵机通常会有一点动差，我们后续会给出算法去补偿这一部分偏差。

# 舵机恢复出厂设置实验

# （Set\_Steer\_Reset）

在这一章，我们将学习把舵机恢复出厂设置，请注意看简介。

5.1 舵机恢复出厂设置实验简介

5.2 舵机恢复出厂设置函数介绍

5.3 Arduino例程文件编写

5.4 下载验证

## 5.1 舵机恢复出厂设置实验简介

恢复出厂设置有啥用？恢复出厂设置顾名思义，就是把舵机的内部寄存器全部恢复到出厂设置，仅此而已。

老问题，怎么把舵机恢复到出厂设置？老回答，我们封装好了，你直接调用就好。

注意：恢复出厂设置后，舵机的ID会被设置为0X01。

## 5.2 舵机恢复出厂设置函数介绍

其实这个函数很简单，我们直接给出函数原型，对于新手，建议你们先去查阅中文手册，以增强你们对函数的熟悉度。

函数原型如下：

void Set\_Steer\_Reset();

使用方法，先初始化一个舵机对象，用该对象直接调用该方法。如下所示：

Steer str5(5, &Serial1); //初始化一个ID为5的舵机对象

Str5. Set\_Steer\_Reset(); //使该舵机恢复出厂设置

再次强调：舵机恢复出厂设置后，舵机的ID被置为0X01。

## 5.3 Arduino例程文件编写

我们打开舵机应用层例程中的steer\_reset\_test程序，我们看到如下所示程序：



图5.3.0

程序的注释写的很清楚，无需赘述。

## 5.4 下载验证

如下图所示找到并打开该示例：



图5.4.0

烧写程序到芯片中并打开串口显示器，如下图所示：

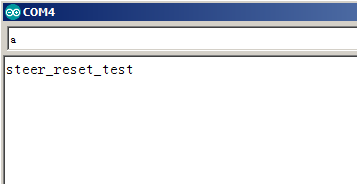


图5.4.1

我们输入任意值并按下回车键，开始恢复出厂设置，结果如下图，代表恢复出厂设置成功：

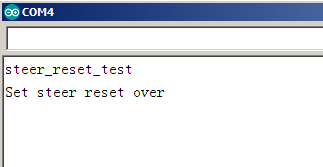


图5.4.2

注意：恢复出厂设置后ID被设为0X01。