$7bot$ 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

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。

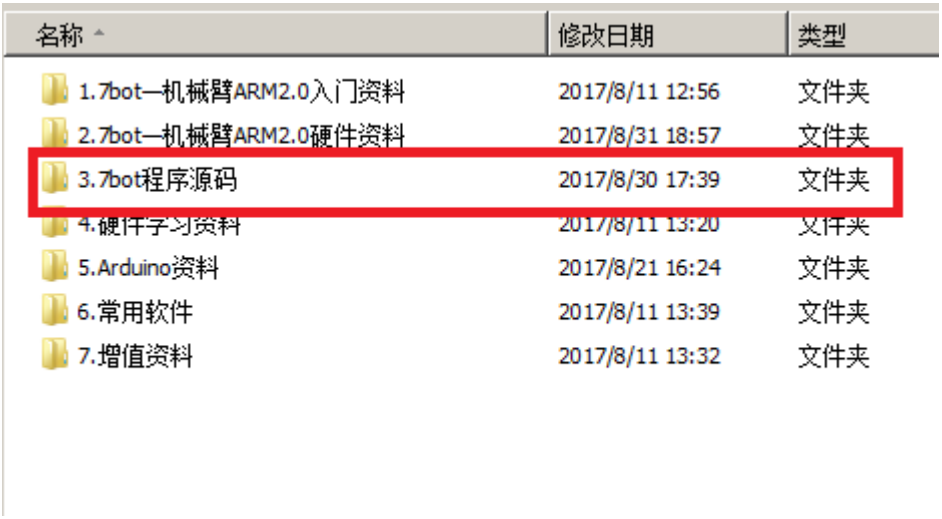

图 0.1

第二步 找到如下所示文件夹，双击打开。

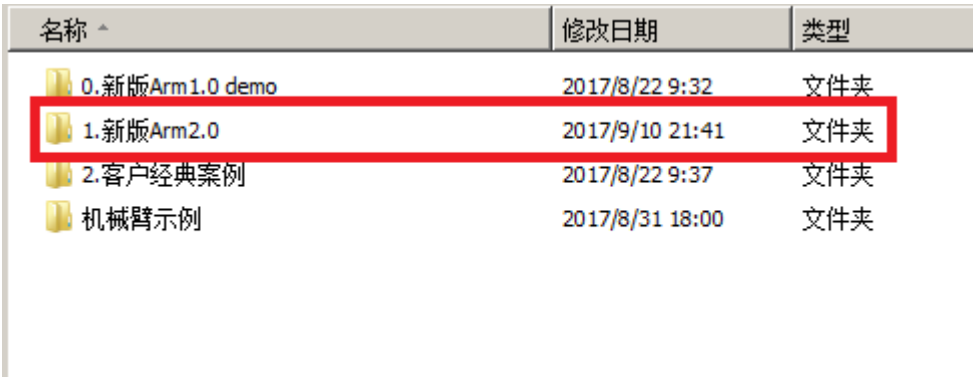

图 0.2

第三步 找到如下所示文件夹，复制。

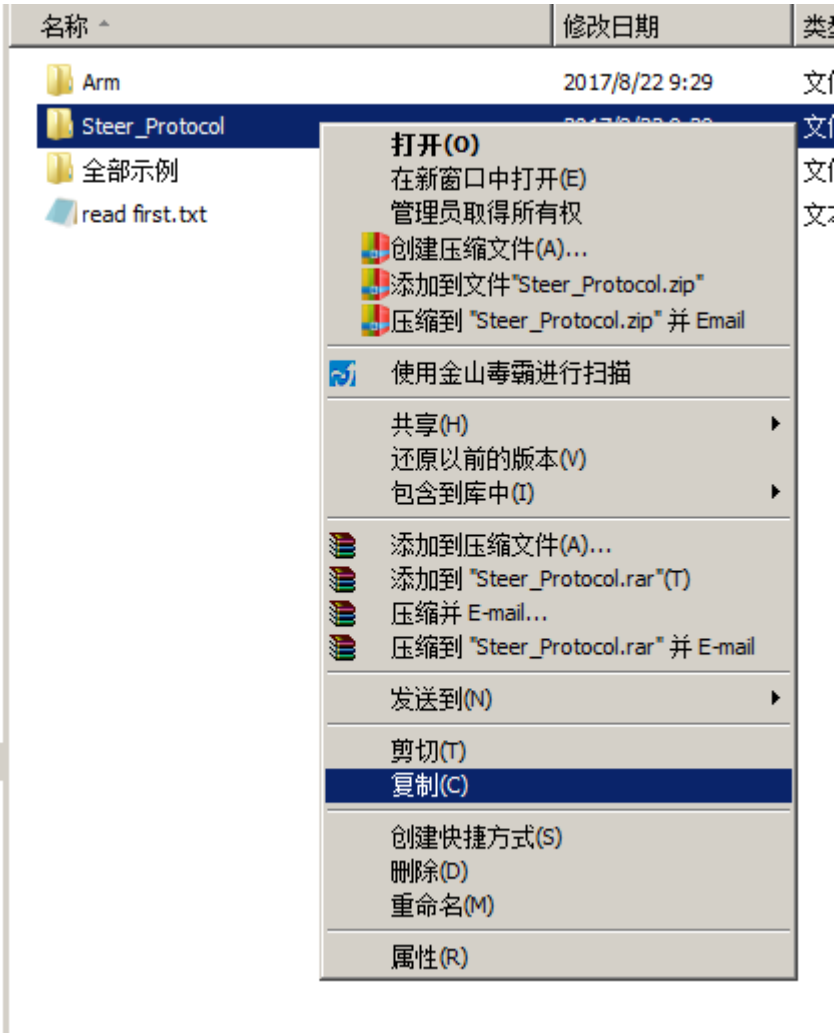

图 0.3

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。

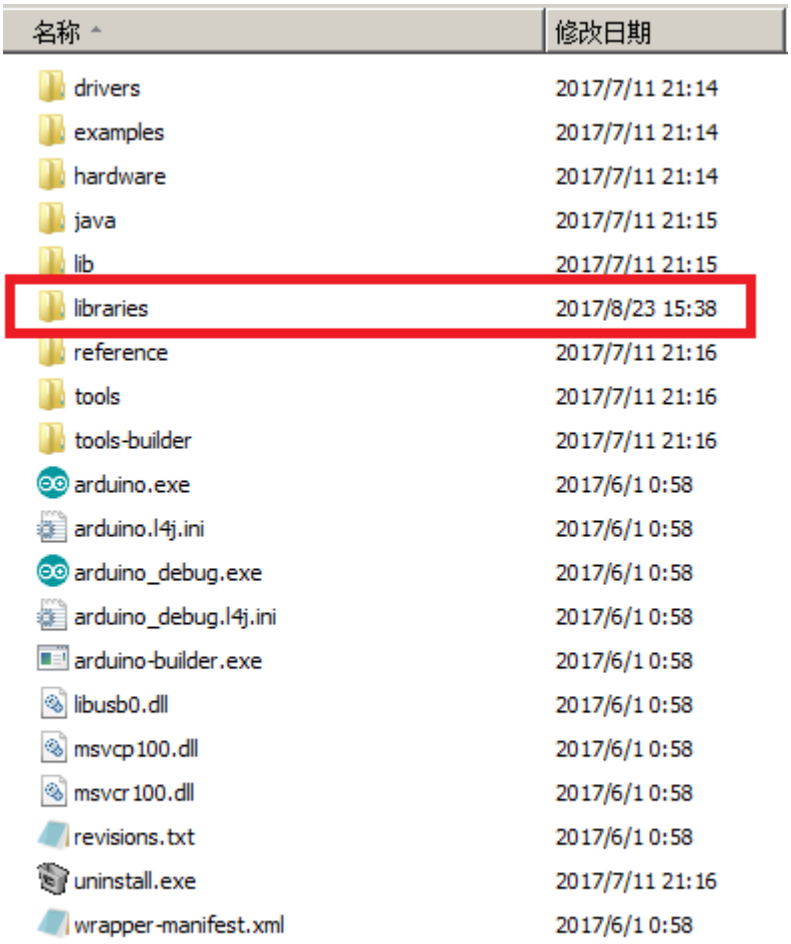

图 0.4

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：

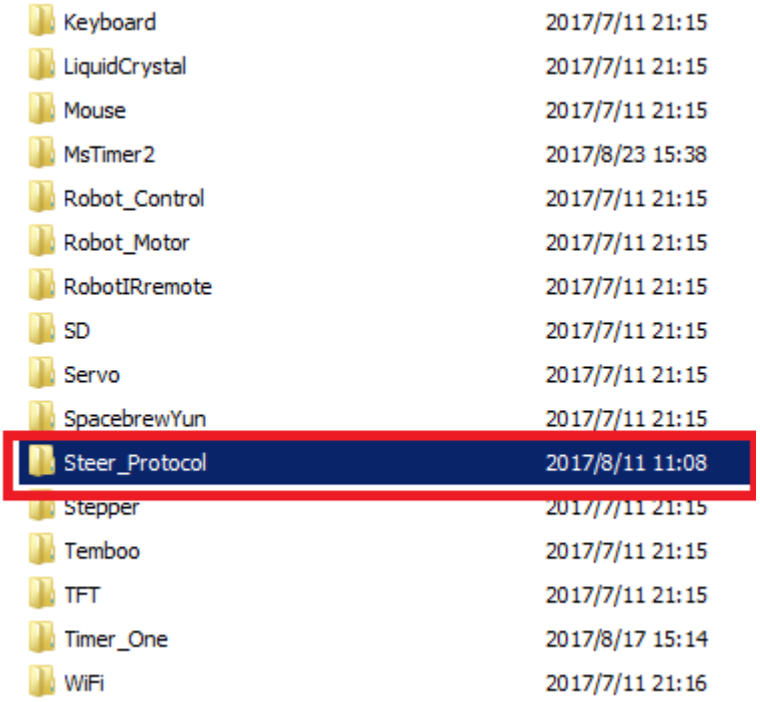

图 0.5

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。


steer_reset_test | Arduino 1.8.3
文件 编辑 项目 工具 帮助
新建 Ctrl+N
打开... Ctrl+O
打开最近的
项目文件夹
示例
关闭 Ctrl+W
保存 Ctrl+S
另存为... Ctrl+Shift+S
页面设置 Ctrl+Shift+P
打印 Ctrl+P
首选项 Ctrl+逗号
关闭 Ctrl+Q
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Examples for any board
Adafruit Circuit Playground
Arm
Bridge
Esplora
Ethernet
Firmata
GSM
LiquidCrystal
MsTimer2
Robot Control
Robot Motor
SD
Servo
SpacebrewYun
Steer_Protocol
Stepper
steer_change_id_test
steer_get_inf_test
steer_get_inf_usetime_test
steer_get_single_inf_test
steer_get_single_inf_usetime_test
steer_move_to_position_test
steer_ping_test
steer_reset_test
上传成功。
项目使用了 2900 字节
全局变量使用了430字节

图 0.6

准备工作完成。

第一章 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。

1.1 舵机测试应答过程简介

1.2 用测试应答为例子去进行中文文档查阅教学

1.3 Arduino 上编写程序

1.4 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

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

图 1.2.1

其次，我们返回目录，在目录中找到相应函数

目录

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

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();
```

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。

```
14 #include<Steer.h>
15
16 Steer mystter1(1,&Serial1);
17 //构造函数输入参数：第一个是ID，第二个是与舵机通信的串口
18
19 void setup() {
20   Serial.begin(115200);
21   Serial.println("Steer_Ping_Test");
22 }
23
24 void loop() {
25   boolean flag = mystter1.Steer_Ping();
26   Serial.print("flag = "); Serial.println(flag);
27   delay(1000);
28 }
```

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：

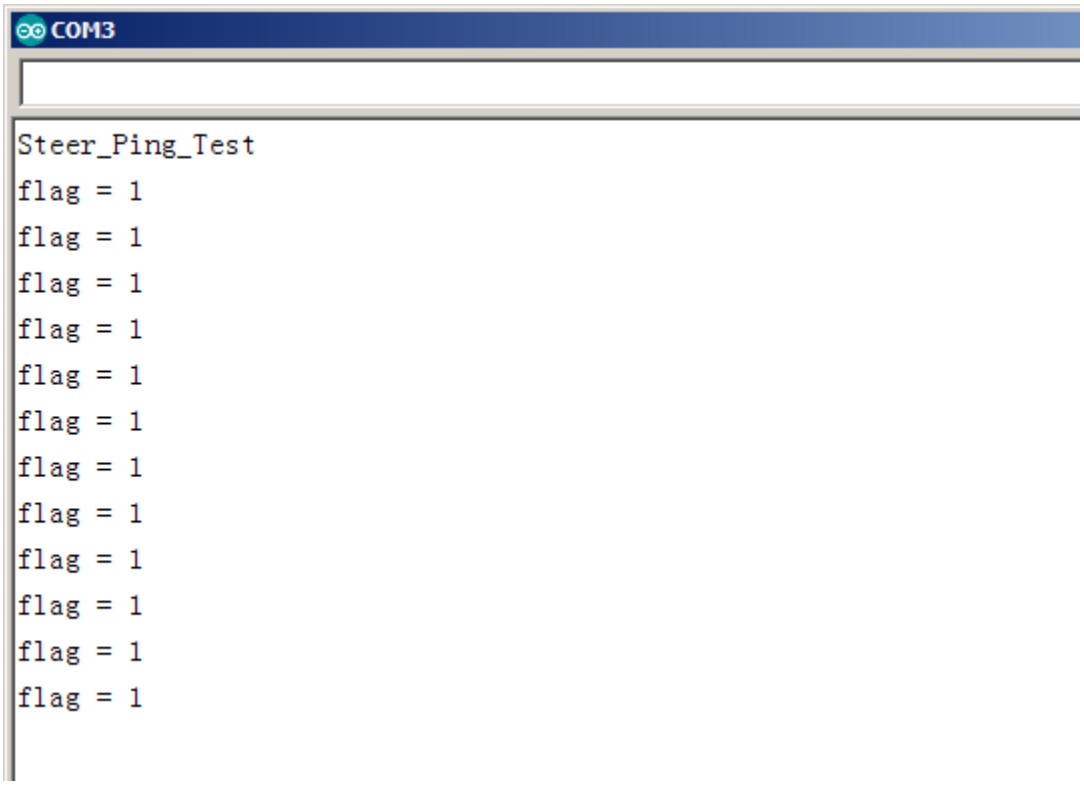

图 1.4.1

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。

第二章 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：

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

4 / 17

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

```
#include<Steer.h>

Steer mystter1(1,&Serial1);      //初始化一个舵机ID为1的对象

void setup() {
  Serial.begin(115200);          //初始化与电脑通信串口（波特率115200）
  Serial.println("steer_change_id_test");
  Serial.println("Please enter in the following format: new_ID");
  Serial.println("such as: 3");
}

void loop() {
   while(Serial.available()){              //等待输入一个新的ID
    byte new_id = Serial.parseInt();       //接收新ID
    mystter1.Change_Steer_ID(new_id);      //改变舵机的ID为新ID
    delay(500);                    //等待改变，必须至少等待500ms（重要）
    if(mystter1.Steer_Ping())              //检验是否改变ID成功
    Serial.println("Change over. Please input another.") ;

    else Serial.println("error: failed to change id");
  }
}
```

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。

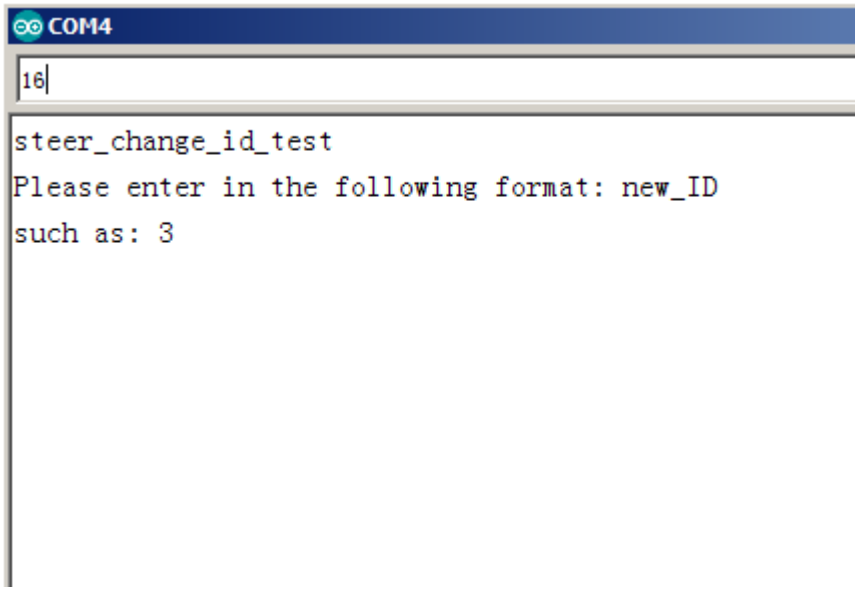

图 2.4.1

按下回车，得到下面结果代表改变 ID 下成功。

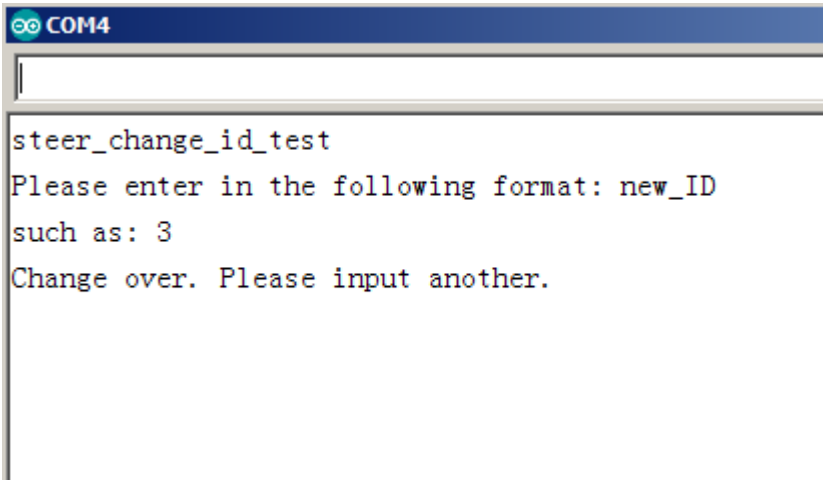

图 2.4.2

我们可以继续输入，继续改变。

第三章 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：

```
#include<Steer.h>

Steer mysteer5(5,&Serial1);        //初始化一个ID为5的舵机对象

void setup() {
  Serial.begin(115200);
  Serial.println("Please input in the following format :");
  Serial.println("position(word)  runtime(word)");
  Serial.println("such as : 2334 4301");
}
int i = 0;                                    //记录次数
void loop() {
   while(Serial.available()){                 //等待接收数据
    word pos = word(Serial.parseInt());       //读取位置数据
    word tim = word(Serial.parseInt());       //读取时间数据
    Serial.println(i++);                      //输出次数
    Serial.print("position = "); Serial.println(pos);
    Serial.print("time = "); Serial.println(tim);
    Serial.println();
    mysteer5.Set_Steer_position_runtime(pos, tim);//控制舵机旋转
  }
}
```

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：

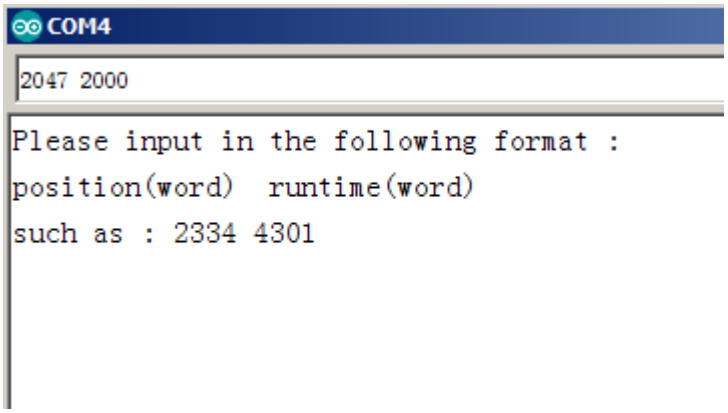

图 3.4.1

按下回车键，听到 ID 为 5 的舵机吱吱吱的开始转了就对了。

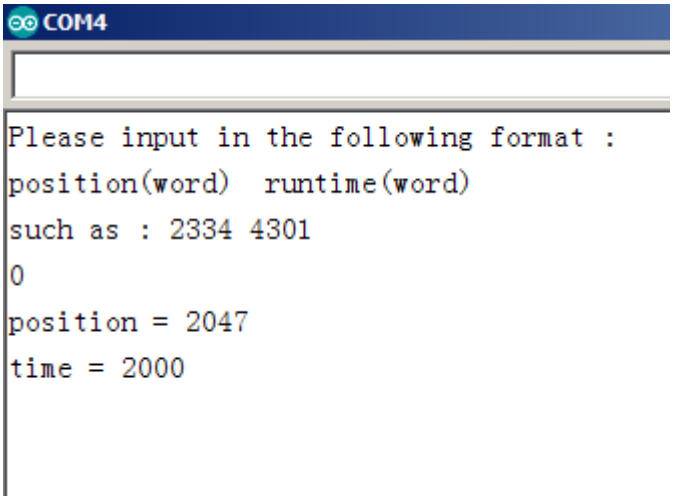

图 3.4.2

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。

第四章 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：

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

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：

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

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：

```
14 #include<Steer.h>
15
16 Steer mysteer5(5,&Serial1);        //初始化一个ID为5的舵机对象
17
18 void setup() {
19   Serial.begin(115200);
20   Serial.println("Steer_Get_Inf_Test");
21 }
22
23 void loop() {
24    while(Serial.available()){     //等待输入去获得信息（任意字符）
25     Serial.read();
26     mysteer5.Get_Steer_All_Inf();//获取全部信息
27     display_all_inf();           //显示全部信息
28   }
29 }
```

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：

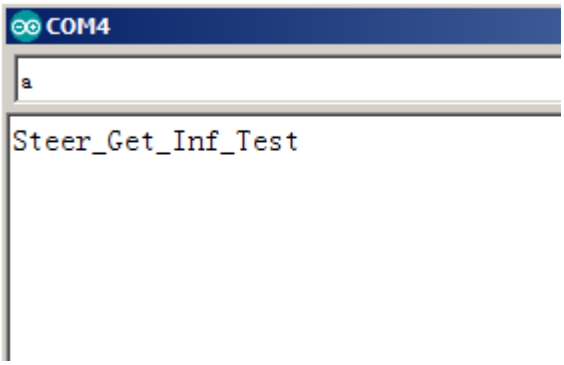

图 4.4.1

按下回车键，读取舵机现在的状态信息，如下如所示：

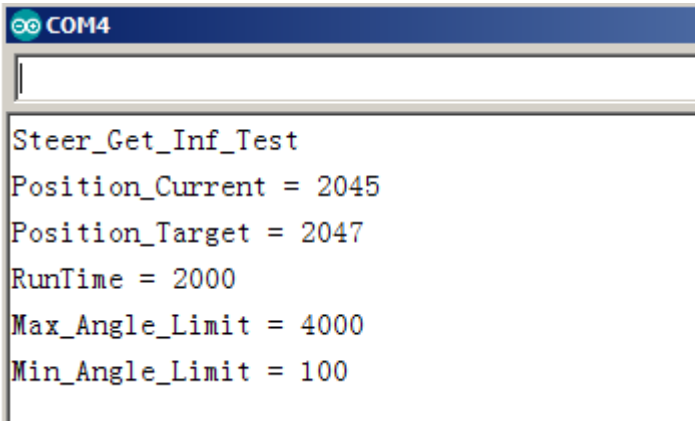

图 4.4.2

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。

第五章 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：

```
#include<Steer.h>
Steer mysteer5(5,&Serial1);

void setup() {
  Serial.begin(115200);
  Serial.println("steer_reset_test");
}

void loop() {
  while(Serial.available())       //按任意键进行复位操作
  {
      Serial.read();
      mysteer5.Set_Steer_Reset();
      delay(500);     //至少延时500ms，等待恢复出厂设置完成
      if(mysteer5.Steer_Ping()) //判断是否恢复出厂设置
      Serial.println("Set steer reset over");
      else Serial.println("Error: Set steer reset error");
  }
}
```

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：

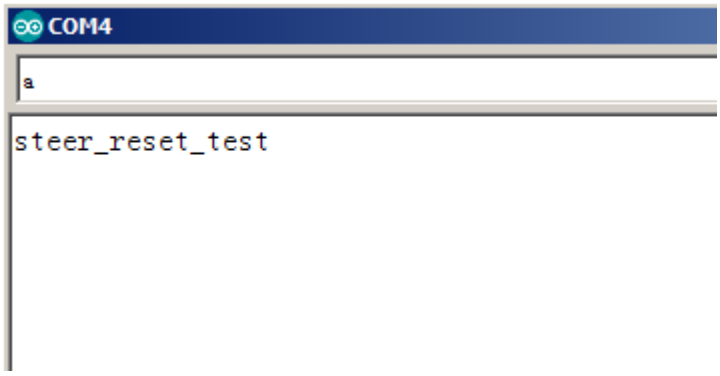

图 5.4.1

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：

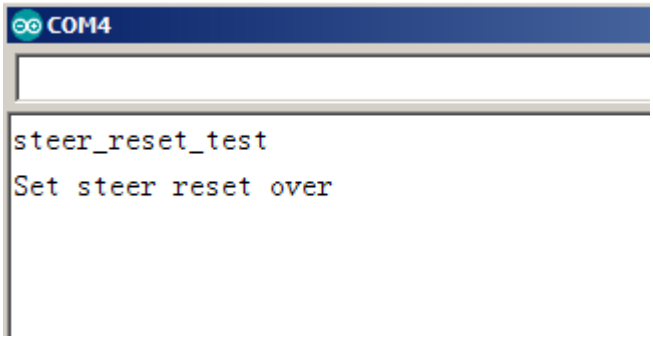

图 5.4.2

注意：恢复出厂设置后 ID 被设为 0X01。