目 录

由于作者水平有限，书中难免有错误和不妥之处，恳请用户予以指证或提出修改意见；
另：本册中的部分内容来自网络，这里对那些不知名的作者表示感谢。

1. 舵机 PWM 信号介绍

1.1 PWM 信号的定义

脉冲宽度调制(PWM)，是英文“Pulse Width Modulation”的缩写，简称脉宽调制，是利用微处理器的数字输出来对模拟电路进行控制的一种非常有效的技术。

PWM 信号的特点在于上升沿与下降沿之间的时间宽度。我们目前使用的舵机主要依赖于模型行业的标准协议，随着机器人行业的渐渐独立，有些厂商已经推出全新的舵机协议，这些舵机只能应用于机器人行业，已经不能够应用于传统的模型上面了。

目前，**亿学通™**的 SG14MD 舵机采用传统的 PWM 协议，它是一款数字型的舵机，其对 PWM 信号的要求较低：

- 不用随时接收指令，减少 CPU 的资源消耗；
- 可以位置自锁、位置跟踪，这方面超越了普通的舵机；

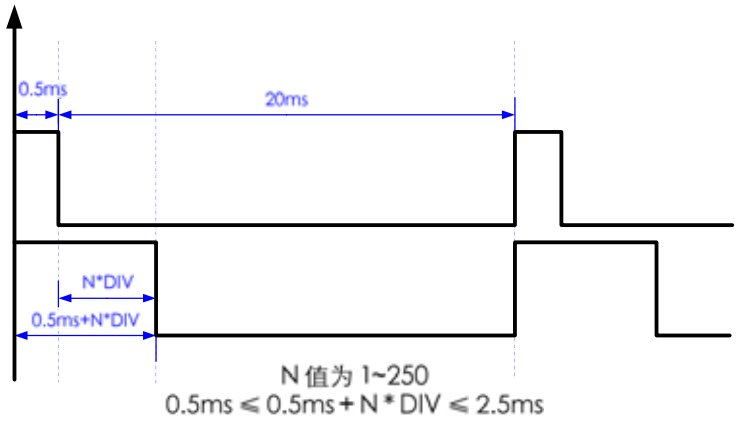


图 1.1 舵机 PWM 控制信号

上图 1.1 是舵机 PWM 控制信号，从前面的介绍，我们可以知道，PWM 信号应该具备以下要求：

- 上升沿与下降沿之间的时间宽度 Tw 最小值应大于 0.5mS，范围为 0.5mS---2.5mS；
- SG14MD 数字舵机对下降沿与上升沿的时间宽度没有要求，如采用 0.5Ms，那么 PWM 就可以是一个周期为 1mS 的标准方波；
- SG06PA 为塑料齿轮模拟舵机，其要求连续供给 PWM 信号；它也可以输入一个周期为 1mS 的标准方波，这时表现出来的跟随性能很好、很紧密。

1.2 舵机控制精度设定

我们采用的是 8 位单片机，其数据分辨率为 256，经过舵机极限参数实验，并为单片机计算保留一些余量，将其划分为 250 份。

从前面介绍可以知道：

脉冲宽度 Tw = 2.5mS-0.5mS= 2mS = 2000uS。

舵机转动角度Φ = 180°

这样我们就能知道 Tw 和Φ是比例对应的。我们将 Tw 和Φ都分成 250 份，那么划分后的每一

份脉冲宽度（我们定义为 DIV）和角度位置也应该是比例对应的。

DIV = 2000uS ÷ 250 = 8uS，所以 PWM 的控制精度为 8us

180 ÷ 250 = 0.72° / DIV ，所以舵机的控制精度为 0.72°

当我们把 PWM 信号的脉冲宽度以 8uS 为单位递增时，舵机转动位置就以 0.72° 进行比例递增，这样我们就可以控制舵机转动角度了。

在 51 单片机时基寄存器中的数值 N ，可以这样设置：

N =（#01H）01 ----（#0FAH）250。

PWM 上升沿宽度 Tw = 0.5mS + N×DIV，从前面的关系可以知道：

0.5mS ≤ 0.5mS + N×DIV ≤ 2.5mS

2. 单舵机拖动及调速算法

2.1 舵机为随动机构

- 当其未转到目标位置时，将全速向目标位置转动；
- 当其到达目标位置时，将自动保持该位置；
  所以对于数字舵机而言，PWM 信号提供的是目标位置，跟踪运动要靠舵机本身；
- 像 SG06PA 这样的模拟舵机需要时刻供给 PWM 信号，舵机自己不能锁定目标位置，
  所以我们的控制系统是一个目标规划系统。

2.1.1 SG14MD 舵机的位置控制方法

舵机的转角达到 180 度，由于采用 8 为 CPU 控制，所以控制精度最大为 256 份。目前经过实际测试和规划，分了 250 份。具体划分参见《250 份划分原理》。同样将 0—180 分为 250 份，每份 0.72 度。

控制所需的 PWM 宽度为 0.5ms—2.5ms，宽度 2ms。2ms÷250=8us；

所以得出：**PWM 信号 = 0.72 度/8us**；

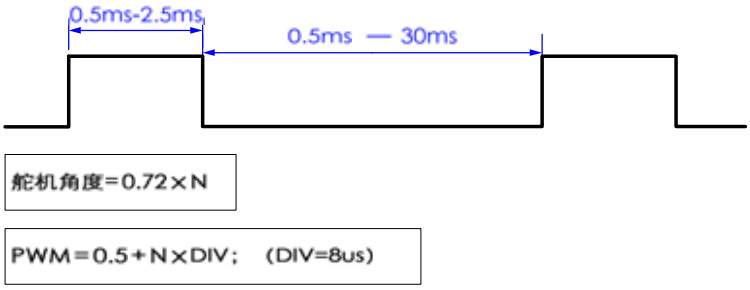


图 2.1PWM 控制信号

角度	0	45	90	135	180
N	0	62	125	187	250
PWM	0.5ms	1ms	1.5ms	2ms	2.5ms

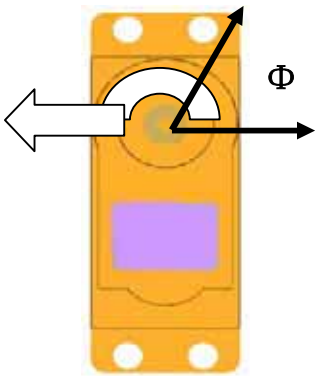
表 2.1 舵机转动角度与 PWM 关系表

2.1.2 SG14MD 舵机的运动协议

舵机的转动方向为：
逆时针为正转

Φ对应 N 值
N=#00H，Φ=0 度
N=#F5H，Φ=180 度

1 ≤ N ≤ 245



运动时可以外接较大的转动负载，舵机输出扭矩较大，而且抗抖动性很好，电位器的线性度较高，达到极限位置时也不会偏离目标。

2.2 目标规划系统的特征

2.2.1 舵机的追随特性

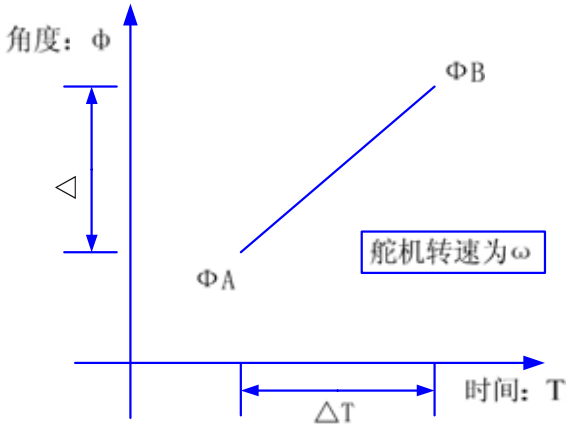


图 2.2 φ-T 坐标

从上图中可以看出：

- 舵机稳定在 A 点不动；
- CPU 发出 B 点位置坐标的 PWM 信号；
- 舵机全速由 A 点转向 B 点：△φ = φB － φA；△T = △φ÷ω；
- CPU 发出 B 点 PWM 信号后，应该等待一段时间，利用此时间让舵机转动至 B 点。

那么，具体的保持（等待）时间如何来计算呢？可根据通过以下方法进行判断：

令：保持时间为 Tk = T －Tw （T 为 PWM 周期，Tw 为上升沿保持时间）：

- 当 Tk ≥△T 时，舵机能够到达目标，并有剩余时间；
- 当 Tk ≤△T 时，舵机不能到达目标；

理论上：当 Tk =△T 时，系统最连贯，而且舵机运动的最快。但实际过程中由于 2 个因素的存在：

- 1 个机器人身上有多个舵机，负载个不相同，所以ω不同；
- 某个舵机在不同时刻的外界环境负载也不同，所以ω不同；

使得连贯运动时的极限△T 难以计算出来。目前采取的方法是经验选取ω值。

2.2.2 舵机ω值测定

舵机的ω值随时变化，所以只能测定一个平均值，或称出现概率最高的值。ω值可以从舵机厂商提供的经验值或者用户自己实际测量中得到。

测试实验：① 将 CPU 开通，并开始延时 Tk；

② 当延时 Tw 到达后，观察舵机是否到达目标；

测定时采用一段双摆程序，伴随示波器用肉眼观察 Tk 与△T 的关系。

2.2.3 舵机ω值计算

一般舵机定为 0.16--0.22 秒/60 度；

我们取较快转速：0.2 秒/60 度 ，360 度用时为 1.2 秒，

于是ω为 360 度/1.2 秒，2Π/1.2 秒

ω=300 度/秒

那么 180 度转动的时间为：

180÷360 度/1.2 秒 = 0.6 秒。

2.2.4 采用双摆试验验证

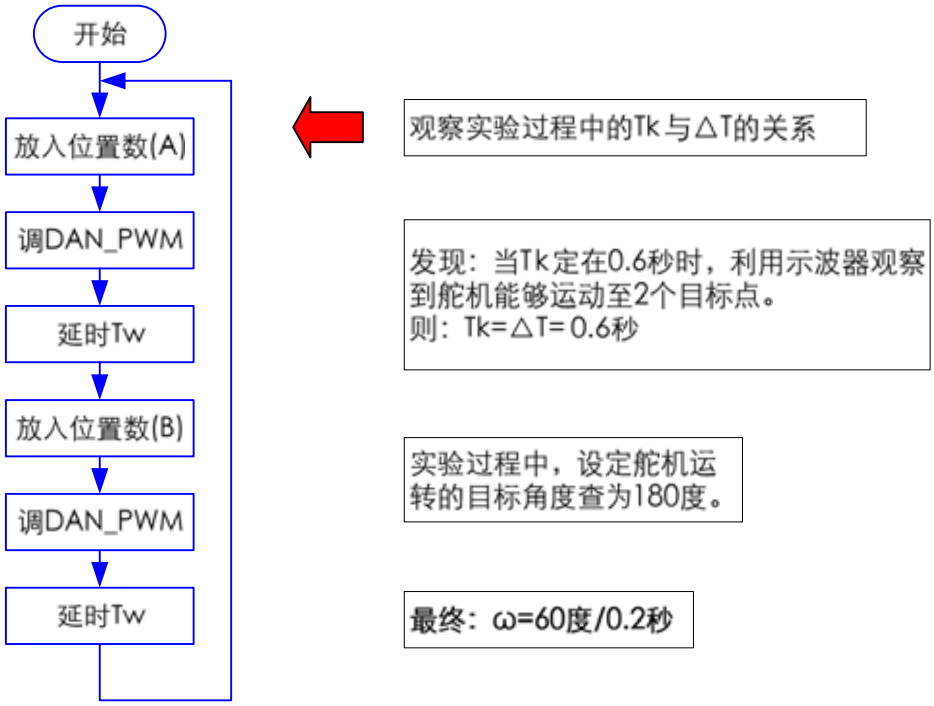


图 2.3 双摆测试流程图

DAV 和 DIV 简介

- DAV 的定义

将 180 度的转角分为 250 个平均小份，则：DAV = 180 度 ÷ 250 = 0.72 度

由于：ω = 0.2 秒/60 度

则：运行 1 DAV 所需时间为，△T 为：0.72 度÷60 度/0.2 秒 = 2.4 mS；

- DIV 的定义

舵机电路支持的 PWM 信号为 0.5mS—2.5mS，总间隔为 2mS。

若分为 250 小份，则 DIV = 2mS÷250 = 0.008 mS = 8uS；

2.5 单舵机调速算法

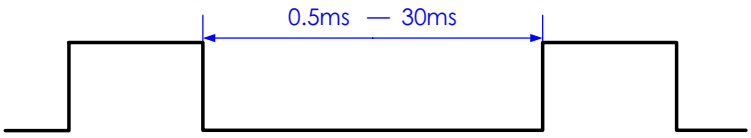


图 2.4 PWM 控制信号波形

测试内容：将后部下降沿的时间拉至 30ms 没有问题，舵机照样工作。

将后部下降沿的时间拉至 10ms 没有问题，舵机照样工作。

将后部下降沿的时间拉至 2.6ms 没有问题，舵机照样工作。

将后部下降沿的时间拉至 500us 没有问题，舵机照样工作。

实践检验出：下降沿时间参数可以做的很小。目前实验降至 500uS，依然工作正常。

原因是：（1）舵机电路自动检测上升沿，遇上升沿就触发，以此监测 PWM 脉宽“头”。

（2）舵机电路自动检测下降沿，遇下降沿就触发，以此监测 PWM 脉宽“尾”。

2.5.1 舵机转动时的极限下降沿 PWM 脉宽

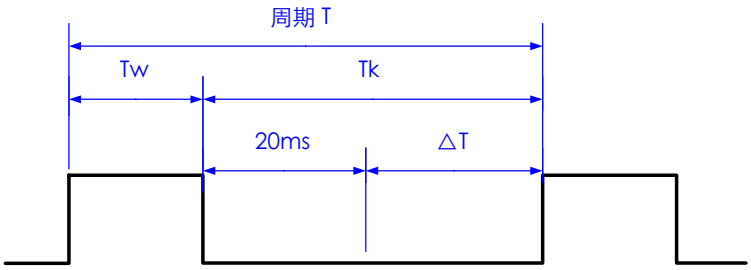


图 2.5 PWM 控制信号波形

△T：舵机运转 1DAV（0.72 度）所需要的最小时间，目前计算出的数值为 2.4mS；

△T 前面的 20 mS 等待时间可以省略，舵机依然工作；而且得出舵机跟随的最快驱动方式。

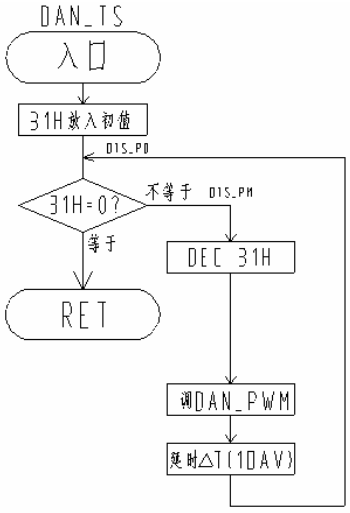


图 2.6 跟随算法流程图

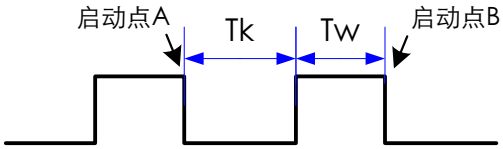
舵机 Tk 数据实验表格

Tk 值	舵机运转特性	Tk 与△T 关系	该程序可行度	备注
500us	不能跟随	Tk < △T	不可行	
800us	不能跟随	Tk < △T	不可行	
1ms	不能跟随	Tk < △T	不可行	
1.1ms	跟随	Tk ≈ △T	可行	最快、平滑
1.2ms	跟随	Tk > △T	可行	最快、平滑
1.6ms	跟随	Tk > △T	可行	最快、平滑
2ms	跟随	Tk > △T	可行	最快、平滑
2.6ms	跟随	Tk > △T	可行	最快、平滑
10ms	跟随	Tk >> △T	可行	较慢、平滑
20ms	跟随	Tk >> △T	可行	较慢、平滑
30ms	跟随	Tk >> △T	可行	较慢、平滑
40ms	跟随	Tk >> △T	可以	较慢、微抖
50ms	跟随	Tk >> △T	可以	很慢、微抖
70ms	跟随	Tk >> △T	不可以	很慢、较抖
100ms	跟随	Tk >> △T	不可以	很慢、较抖

令人质疑的地方为 1.1ms 时的表现，得出的 Tk ≈ △T；

也就是说 1.1ms ≈ 2.4ms，这显然存在问题。

经过考虑重新观察 PWM 波形图发现，电机真正的启动点如下图：



极限转动方式

实际上由 A 到 B 的运动时间为：

△T = Tk +Tw = Tk+ 0.5mS + 1×DIV

3. 舵机联动单周期 PWM 指令算法

3.1 控制要求

要求同时发给 8 个舵机位置目标值，该指令的执行周期尽量短，目的有 2 个：其一，是为了将来扩充至 21 个舵机；其二，目标越快，舵机的转动速度越快；

我们以 8 路为 1 组或称 1 个单位，连续发出目标位置，形成连续的目标规划曲线，电机在跟随过程中自然形成了位置与速度的双指标曲线，实现 8 路舵机联动。

3.2 注意事项

从 24 个端口，P0.0、P1.0 到 P2.0，单 DIV 循环的最小时间只有 8us，所以串行运算是不行的，那么就采用并行运算。

目前采用的并行算法是 P0.0—P0.7 为一个基本单位，8 位一并。

实际案例：P1 口的 8 个位置个不相同；

端口	N 寄存器	目标位置（度）	N 数值（整数）	Tw 时间宽度（ms）
P1.0	position[0]	0	0	0.500
P1.1	position[1]	0.72	1	0.508
P1.2	position[2]	45	62.5	1.000
P1.3	position[3]	50	69.4	1.055
P1.4	position[4]	60	83.3	1.166
P1.5	position[5]	90	125	1.500
P1.6	position[6]	135	187.5	2.000
P1.7	position[7]	180	250	2.500

注意：N 要取整数；所以不能精确实现的有 45 度、50 度、60 度、135 度等位置

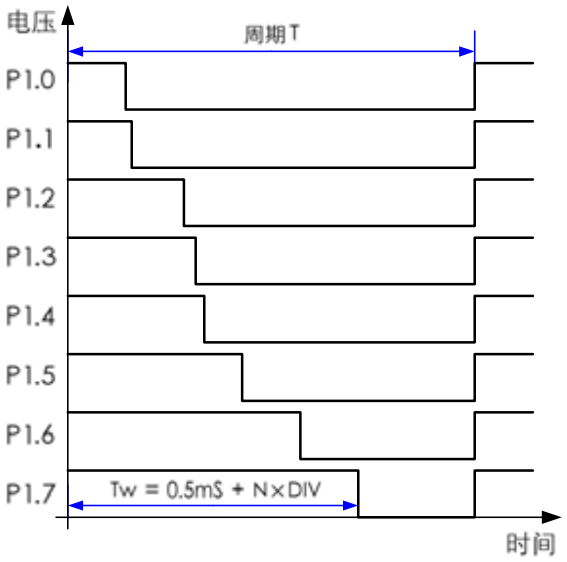


图 3.1 P1 口 PWM 控制信号

3.3 8 路 PWM 信号发生算法解析

我们预计将整个周期控制在 3.5-5ms 内；由上图得知：P1 口的 8 个端在不同时间产生下降沿。

那么由上例如：我们的 P1.5 口，他的 N 为 125；那么就需要它在 125 个 DIV 后产生下降沿，时间为（125*8us=1000us）。

我们在其中发现 2 个关键参数：①时间参数 N=125　②逻辑参数 P1.5=#0DFH

逻辑参数的定义：如下，采用&指令，操作 P1 口。

	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	备注
P1.0= # FEH	1	1	1	1	1	1	1	0	
P1.1= # FDH	1	1	1	1	1	1	0	1	
P1.2= # FBH	1	1	1	1	1	0	1	1	
P1.3= # F7H	1	1	1	1	0	1	1	1	
P1.4= # EFH	1	1	1	0	1	1	1	1	
P1.5= # DFH	1	1	0	1	1	1	1	1	
P1.6= # BFH	1	0	1	1	1	1	1	1	
P1.7= # 7FH	0	1	1	1	1	1	1	1	

例如：将 P1.5 口产生下降沿，就将# 0DFH 去“&” P1 口。

逻辑“&”指令，逢“0”得“0”，不影响其他位。

具体的程序操作如下：

① 开 3.5ms 定时中断
② 取出 8 个端（P1.0-P1.7）的位置值，也就是 8 个 N 值；并赋予相应的端逻辑参数；
③ 将这 8 个值由大到小排列，相应端的逻辑参数值也随着 N 的顺序排列，一一对应；
④ 将 N 值做减法，求得：
M1=N1
M2=N2-N1
M3=N3-N2
M4=N4-N3
M5=N5-N4
M6=N6-N5
M7=N7-N6
M8=N8-N7
⑤ 取出 M1，延时 M1*DIV，& 相应的逻辑参数；
取出 M2，延时 M2*DIV，& 相应的逻辑参数；
取出 M3，延时 M3*DIV，& 相应的逻辑参数；
取出 M4，延时 M4*DIV，& 相应的逻辑参数；
取出 M5，延时 M5*DIV，& 相应的逻辑参数；
取出 M6，延时 M6*DIV，& 相应的逻辑参数；
取出 M7，延时 M7*DIV，& 相应的逻辑参数；
取出 M8，延时 M8*DIV，& 相应的逻辑参数；
⑥ 8 个端的下降沿全部产生完毕，等待一定的 Tw 值，或等待 3.5ms 中断的到来；
⑦ 中断到来后，清理中断标志，然后结束该程序。

注意事项：当进行逐个排序延时的过程中，CPU 要取出 M1、M2、M3…M8，那么会有 1 个取数指令周期，当 CPU 采用 12MHz 时为 1us。最终应该在第 8 个延时，即 M8 时扣除掉，具体指令参见指令集。

3.4 N 排序子程序 RAM 的制定

入口处

端口	N 值寄存器地址	&逻辑数寄存器地址	&逻辑数值
P1.0	position[0]	kouchu[0]	#FEH
P1.1	position[1]	kouchu[1]	#FDH
P1.2	position[2]	kouchu[2]	#FBH
P1.3	position[3]	kouchu[3]	#F7H
P1.4	position[4]	kouchu[4]	#EFH
P1.5	position[5]	kouchu[5]	#DFH
P1.6	position[6]	kouchu[6]	#BFH
P1.7	position[7]	kouchu[7]	#7FH

备注：position[7]寄存器内存放的是 P1.7 端口的 N 值；kouchu[7]寄存器内存放的是 P1.7 端口的&逻辑参数值；

出口处

从左到右为 N 值从大到小排列　　（大 ＞N 值 ＞ 小）

N 值寄存器地址	&逻辑数寄存器地址	&逻辑数值
paixu_ncha[0]	kouchu[0]	未知
paixu_ncha[1]	kouchu[1]	未知
paixu_ncha[2]	kouchu[2]	未知
paixu_ncha[3]	kouchu[3]	未知
paixu_ncha[4]	kouchu[4]	未知
paixu_ncha[5]	kouchu[5]	未知
paixu_ncha[6]	kouchu[6]	未知
paixu_ncha[7]	kouchu[7]	未知

所谓的“未知”：由于排列按照大到小顺序，“未知”内存放的为端口信息要根据排序做相应的调整。

备注：paixu_ncha[0]内存放的是某位的 N 值，其值最大；

paixu_ncha[7]内存放的是某位的 N 值，其值最小；

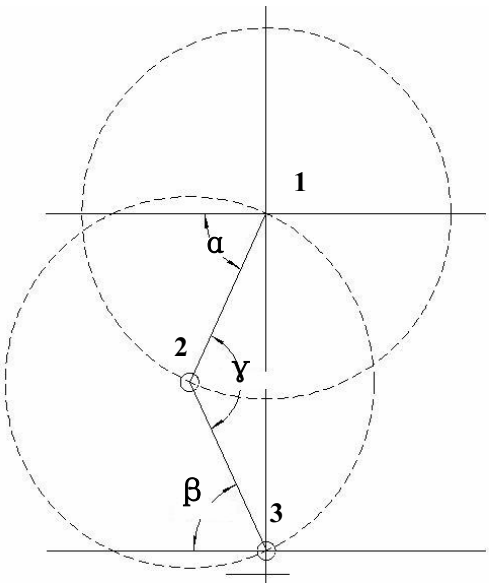
kouchu[0]—kouchu[7]内存放数，可以根据其数值判断出是具体那个端口的下降沿。

例如：其值为“0xFB”那么它就是 P1.2；

4. Roboy 左腿机械结构

4.1 左腿部侧面轴关节介绍

双足机器人行走和它的机械结构有至关重要的联自从 1986 年以来,一直延用的日本加藤一郎结构,其腿部侧面就如右图所示:它有 3 个自由度,每个自由度采用 1 个舵机构成。图中用 1、2、3 表示这 3 个舵机。



其中包含几个重要的几何关系:

一般的机器人，腿上半部与下半部长短相近，我们在研究机器人步伐的时候可以令其为相等长度。这是 1 个十分关键的地方。在国内，之所以大多的机器人不能行走，其中的一个原因就是将腿上半部与下半部加工成不同长度。导致 CPU 的计算量剧增，国内的数控技术本来就落后,缺少这方面的人才,其他稍微会些软件就自命不凡的人站出来就搞,结果连最基本的第一关都不能过,根本不能进行下面的研究。

注意：1-2 和 2-3 的长短要一致会骤减 CPU 的计算量!

原地踏步动作时:

∵L12=L23

∴△α=△β

∴△γ=△α+△β=2△α

（如此简单的函数关系，可以骤减 CPU 的计算量）

今后在做一般性研究时，可以将 L12 和 L23 做成任意长度。

如此描述一个简单的微分函数(原地踏步函数):

∵dα=dβ

∴dγ= dα+dβ（一般 CPU 就用此式进行积分运算）

取△α为 1DAV，则△β=1DAV，△γ=2DAV

即：舵机 1 的|△N|=1

舵机 2 的|△N|=2

舵机 3 的|△N|=1

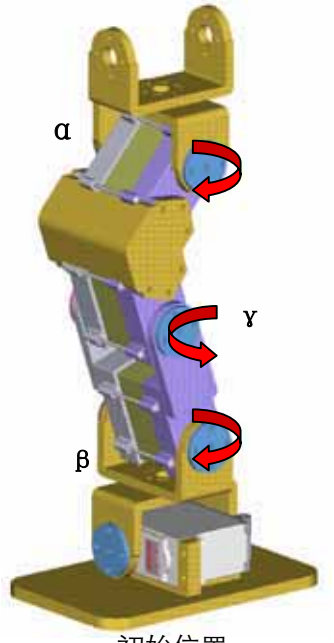
CPU 一边进行积分运算，一边将数据发送给舵机，令其执行。

∴∫dγ=∫(dα+ dβ)=∫dα+∫dβ

∴Kγ+△γ=（Kα+△α）+（Kβ+△β）

Kγ、Kα、Kβ为积分原始初始值，在机器人中表现为初始位置坐标。

4.2 舵机转动正方向制定



初始位置

初始位置
α=169
β=91
γ=112

我们研究腿部运动时，按照从初始位置（即最高站立姿式）起向最低站立姿势过渡时的舵机转动方向为正方向。那么图中标出了 3 个舵的正方向。

由于舵机规定的正方向为逆时针方向，则：
1．方向 1 与舵机同向
2．方向 2 与舵机同向
3．方向 3 与舵机反向

（1）高位到低位为上半周；影响微分函数

（2）低位到高位为下半周；影响微分函数

上半周：舵机 1 的 △N = +
舵机 2 的 △N = +
舵机 3 的 △N = -

下半周：舵机 1 的 △N = -
舵机 2 的 △N = -
舵机 3 的 △N = +

4.3 左腿半周期运动程序分析

依照上面所讲述的 2 个几何特点，① L12=L23；

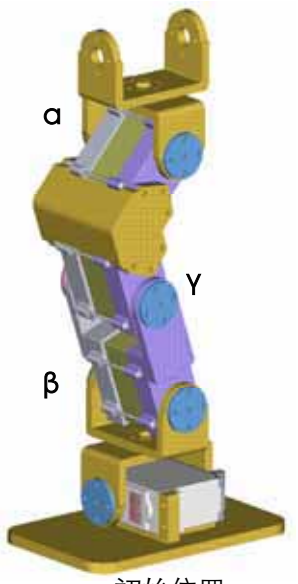
② 1、3 点始终垂直共线；

∴ 舵机 1 的|△N|=1

舵机 2 的|△N|=2

舵机 3 的|△N|=1

CPU 一边进行积分运算，一边将数据发送给舵机。



初始位置

初始时刻
α= 169
γ= 91
β= 112

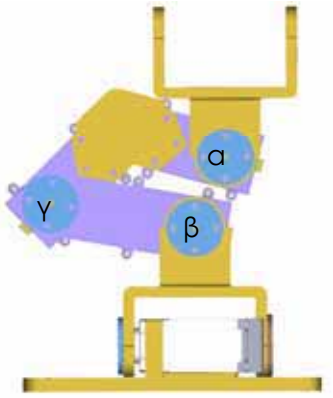
上半周：舵机 1 的 △N = +1
舵机 2 的 △N = +2

下半周：舵机 1 的 △N = -1
舵机 2 的 △N = -2
舵机 3 的 △N =

CPU 进行上半周积分时，腿部逐渐下蹲；
CPU 进行下半周积分时，腿部逐渐上抬；

积分始：①α=169
②γ=91
③β=112

积分末：①α=245
②γ=243
③β=36



积分末位置

至积分末，发现α先出现极限值#0F5H，那么积分步数为：245-169=76 步。
76DAV=76×0.74 度=56.24 度
∴△α=56.24 度
△γ=112.48 度
△β=56.24 度

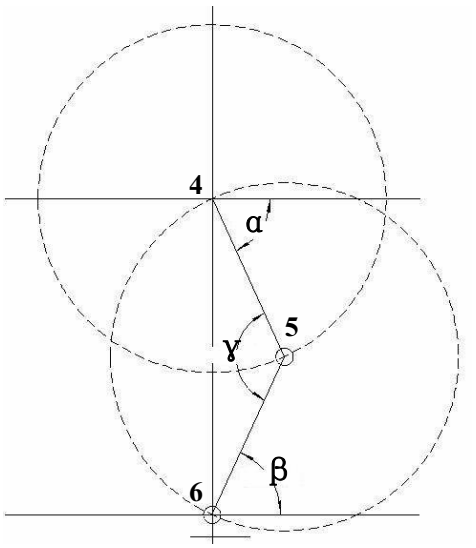
以上介绍的是积分上半周的动作系数

5. Roboy 右腿机械结构

5.1 右腿部侧面轴关节介绍

如同左腿一样，右腿的机械结构与其形成平面对称结构。属于日本加藤一郎结构。

其腿部侧面就如右图所示:它有 3 个自由度,每个自由度采用 1 个舵机构成。图中用 4、5、6 表示这 3 个舵机。



同样包含几个重要的几何关系:

一般的机器人，腿上半部与下半部长短相近，我们在研究机器人步伐的时候可以令其为相等长度。这是 1 个十分关键的地方。

在计算过程中，令腿的上半部与下半部长度相等。

原地踏步动作时:

∵L45=L56

∴△α=△β

∴△γ=△α+△β=2△α

（如此简单的函数关系，可以骤减 CPU 的计算量）

注意：4-5 和 5-6 的长短要一致会骤减 CPU 的计算量!

今后在做一般性研究时，可以将 L45 和 L56 做成任意长度。

如此描述一个简单的微分函数(原地踏步函数):

∵dα=dβ

∴dγ= dα+dβ（一般 CPU 就用此式进行积分运算）

取△α为 1DAV，则△β=1DAV，△γ=2DAV

即：舵机 4 的|△N|=1

舵机 5 的|△N|=2

舵机 6 的|△N|=1

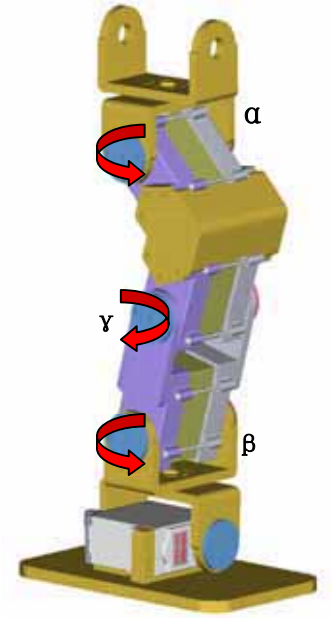
CPU 一边进行积分运算，一边将数据发送给舵机，令其执行。

∴∫dγ=∫(dα+ dβ)=∫dα+∫dβ

∴Kγ+△γ=（Kα+△α）+（Kβ+△β）

Kγ、Kα、Kβ为积分原始初始值，在机器人中表现为初始位置坐标。

5.2 舵机转动正方向制定



初始位置

初始位置
α=80
β=158
γ=128

我们研究腿部运动时，按照从初始位置（即最高站立姿式）起向最低站立姿势过渡时的舵机转动方向为正方向。那么图中标出了 3 个舵的正方向。

由于舵机规定的正方向为逆时针方向，则：
1．方向 4 与舵机反向
2．方向 5 与舵机反向
3．方向 6 与舵机同向

（1）高位到低位为上半周；影响微分函数

（2）低位到高位为下半周；影响微分函数

上半周：舵机 4 的 △N = -
舵机 5 的 △N = -
舵机 6 的 △N = +

下半周：舵机 4 的 △N = +
舵机 5 的 △N = +
舵机 6 的 △N = -

5.3 右腿半周期运动程序分析

依照上面所讲述的 2 个几何特点，① L45=L56；

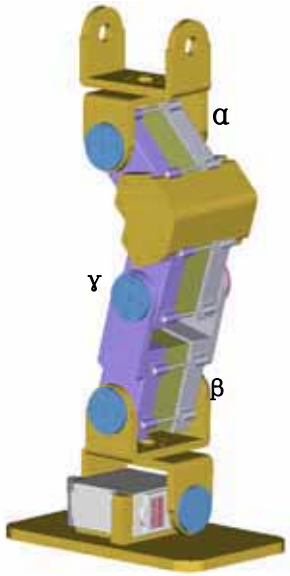
② 4、6 点始终垂直共线；

∴ 舵机 4 的 | △N | =1

舵机 5 的 | △N | =2

舵机 6 的 | △N | =1

CPU 一边进行积分运算，一边将数据发送给舵机。



初始位置

初始时刻
α= 80
γ= 158
β= 128

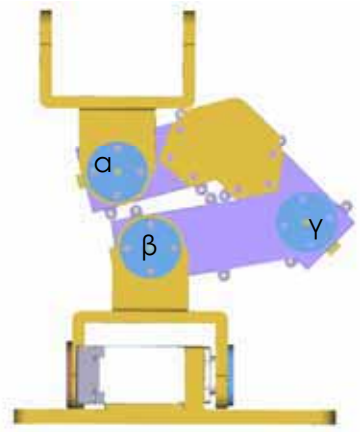
上半周：舵机 4 的 △N = -1
舵机 5 的 △N = -2
舵机 6 的 △N =

下半周：舵机 4 的 △N = +1
舵机 5 的 △N = +2

CPU 进行上半周积分时，腿部逐渐下蹲；
CPU 进行下半周积分时，腿部逐渐上抬；

积分始：① α =80
② γ =158
③ β =128

积分末：① α =4
② γ =6
③ β =204



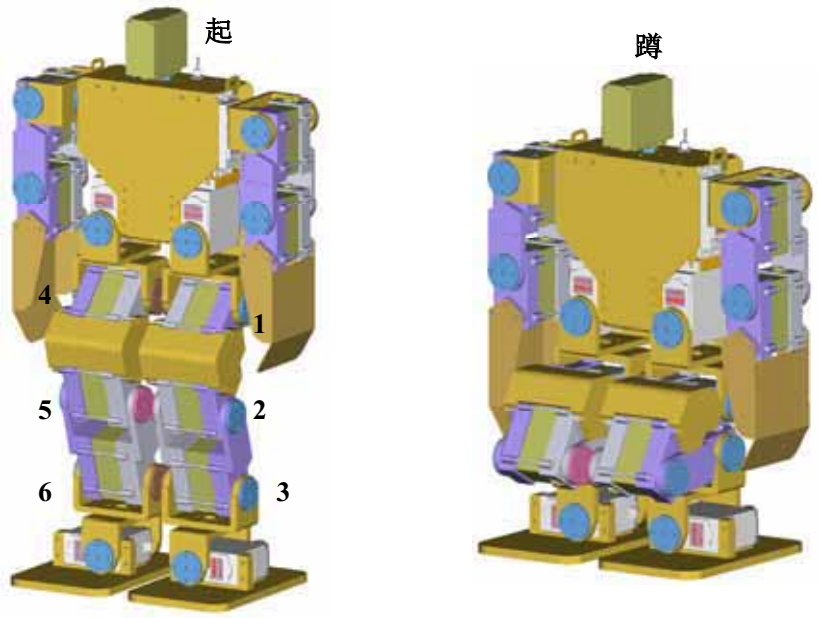
积分末位置

至积分末，发现α先出现极限值#0F5H，那么积分步数为：204-128=76 步。
76DAV=76×0.74 度=56.24 度
∴ △α=56.24 度
△γ=112.48 度
△β=56.24 度

以上介绍的是积分上半周的动作系数

6. 双腿蹲起动作解析

6.1 双腿蹲起动作程序特点



相对其他程序，蹲起程序较为简单，但恰恰是最基础的程序，所以在所有程序中处于中流砥柱的位置。在机器人蹲起过程中

有以下 4 个特点：①双腿着地；

②同起同落；

③脚、肩保持平行；

④胸平面轴不动；

根据这 4 个特点，我们采用积分程序将双腿的运动过程中的各个关节位置计算出来，并实时输送给舵机。共控制 6 个运动轴，分别为左腿的 1、2、3 和右腿的 4、5、6。

其中：△α=△1=△4

△γ=△2=△5

△β=△3=△6

△α=△β

△γ=2△α

规定蹲为动作的上半周期

规定起为动作的下半周期

舵机与机器人结构存在相位差,具体注释如下:

1—同相 2—同相 3—反相	4—反相 5—反相 6—同相

6.2 积分首末位置与过程

我们将积分过程分为上下 2 个半周期，则每条腿有 4 个初始位置。

即：A 积分至 B
B 积分至 A

上半周：

1.	169	245
2.	91	243
3.	112	36

4.	80	4
5.	158	6
6.	128	204

下半周：

1.	169	245
2.	91	243
3.	112	36

4.	80	4
5.	158	6
6.	128	204

最多可以积分 76 步，关节 1、4 会到达极限值。

积分**起始**位置可以改变（条件 1、2、3；4、5、6 遵守肩、脚平行）

积分**结束**位置可以改变（条件 1、2、3；4、5、6 遵守肩、脚平行）

起始位置公式：

1= 169+△N	4= 80 - △N
2= 91 + 2△N	5=158 - 2△N
3=112 -△N	6=128 + △N

结束位置公式：

1= 245 -△N	4= 4+ △N
2= 243- 2△N	5=158 + 2△N
3=36 + △N	6=128 - △N

6.3 积分中间位置

积分的中间位置是个特殊位置，在这点上双腿的运动幅度占总幅度的一半，那么就对将来的原地踏步动作起着关键作用。根据几何关系得出在踏步运动中，积分中点是左、右脚的换脚位置，顾十分重要；另外，在随时调节踏步的幅度时，中点的漂移与控制是个值得关注的技术，以下逐步分析中点坐标的计算方法。

例：当 1 的起始值为 169，1 的结束值为 245 时，

积分全过程　　△N=#4CH=76 步

几分半过程　　1/2△N=38 步

则中点坐标为：1=207

2=167

3=74

当 4 的起始值为 80，1 的结束值为 4 时，

积分全过程　　△N=76 步

几分半过程　　1/2△N=38 步

则中点坐标为：4=42

5=82

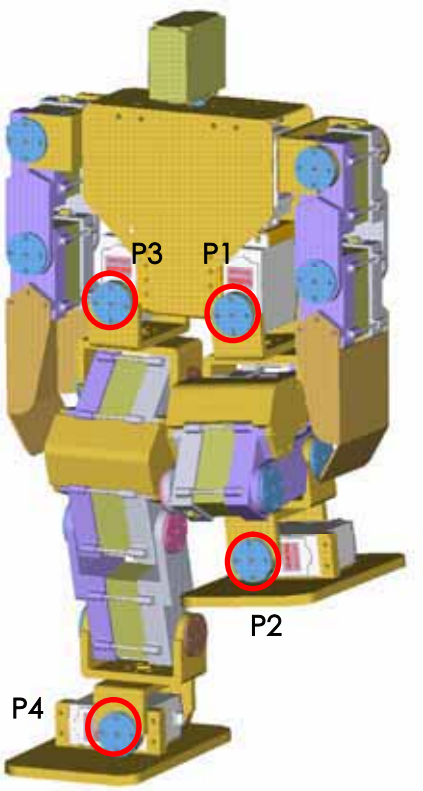
6=166

中点坐标公式为

M1 = 1 起始 + 1/2△N M2 = 2 起始 + △N M3 = 3 起始 - 1/2△N	M4 = 4 起始 - 1/2△N M5 = 5 起始 - △N M6 = 6 起始 + 1/2△N

7. P 平面动作舵机

7.1 新增 P 平面运动舵机介绍



新增的胸平面舵机有 4 个，分别为：左腿的 P1、P2；右腿的 P3、P4。

其中如果保持肩、脚平行，身体不倾斜，那么：
左腿的 P1、P2 为平行关系；
右腿的 P3、P4 为平行关系；

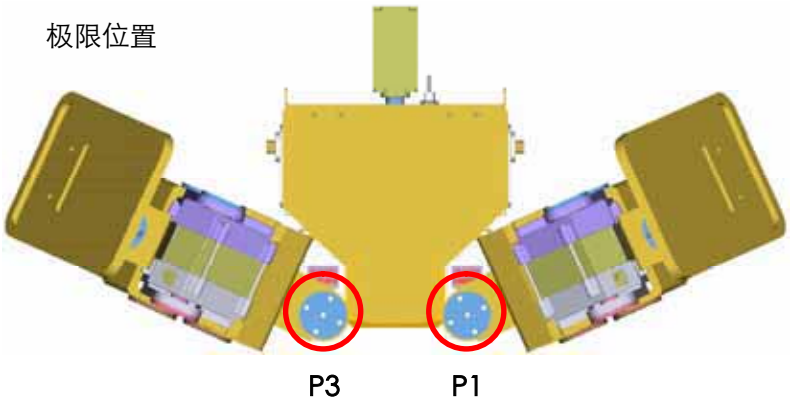
特别注意，当形成周期运动，且保持肩、脚平行，身体不倾斜，那么：
P1、P2、P3、P4 全为平行关系；
即：△P1=△P2=△P3△P4

初始位置：
P1=104
P2=104
P3=146
P4=146

该位置参数来自 1 台 RB-2 实验机，其他机器人参数根据各自舵机初始位置而定

7.2 P1、P2、P3、P4 舵机的初始位置

我们就以机器人直立姿势为初始位置，本应该为舵机的固有机械中点即电位器中点，但考虑到 RB-2 机器人的特殊结构设计也就是侧滚翻动作的完成，那么如下图的极限位置：



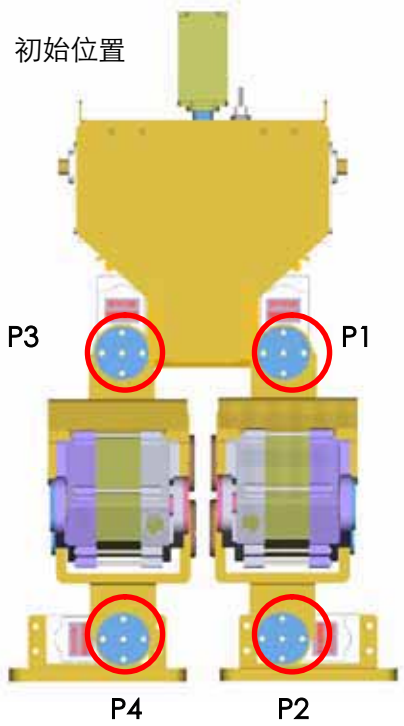
极限位置：
P1=245
P2=245
P3=0
P4=0

规定：机器人逐渐进行劈叉动作时的运动方向为正方向。

则：P1 同相　　P3 反相
　　P2 同相　　P4 反向

注意！实际的 P2 和 P4 两个舵的角度不能达到#0F5H 和#00H，因为机械极限不能达到如此大的角度。否则会破坏机械结构件。但是，为了减少 CPU 的计算量，我们将 P2 与 P4 的值紧紧跟随 P1 与 P3，这样避免了许多初始位置补偿系数。

实际安装过程中要考虑到舵盘的最小机械分度，所以不一定能够做到 P1 与 P2 或 P3 与 P4 初始位置相同，所以还需设定不同的初始值，但相差不会过 5 个点。



初始位置：
P1=104
P2=104
P3=146
P4=146

8. 劈叉动作解析

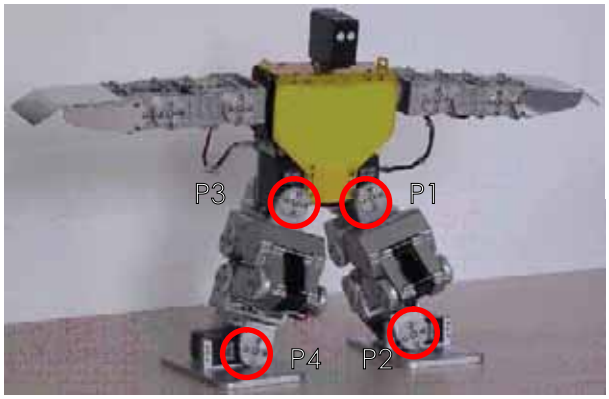
8.1 劈叉动作舵机的方向制定

由于机器人逐渐进行劈叉动作时的运动方向为正方向，那么劈叉动作的 4 个关节上半周全部为正向转动，相应的 4 个舵机的转动方向为：

上半周：P1= +　　P3= -
　　P2= +　　P4= -

下半周：P1= -　　P3= +
　　P2= -　　P4= +

劈叉动作运动的 4 个舵机被安排在 P2 口，所以调用单口调速程序就可以。P1 口不用参与。
注意！安装模拟舵机时需要实时控制其它端口，否则舵机将失去动力。



舵机号	P1	P2	P3	P4
位置寄存器	Position[8]	Position[9]	Position[10]	Position[11]
端口	P2.0	P2.1	P2.2	P2.3
上极限	104	104	146	146
中间态	104	104	146	146
下极限	160	160	90	90

以上数据取自一台实验机器人，其他机器人的初始位置受舵机影响，有所不同。

8.2 劈叉动作的速度与加速度

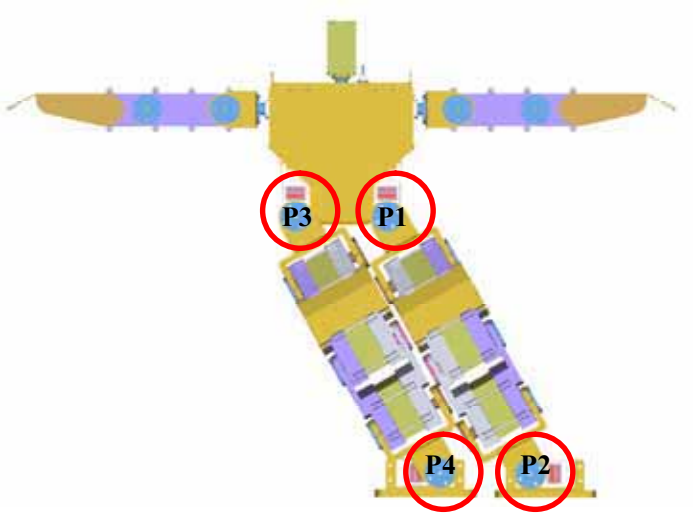
由于劈叉动作属于双腿支撑动作中最稳定的一种，所以可以支持任何一种运动形式。包括急速启动、急速停止、匀速启动或者某种周期函数。

在此，我们暂时不作速度与加速度的额外研究。

9. 双腿侧身动作解析

9.1 动作过程中各舵机的方向制定

由于机器人逐渐进行侧身动作时的运动方向为正方向，那么侧身动作的 4 个关节上半周方向为 2 正 2 反，相应的 4 个舵机的转动方向为：



上半周：P1= +　　P3= +
　　P2= +　　P4= +

上半周：P1= -　　P3= -
　　P2= -　　P4= -

舵机号	P1	P2	P3	P4
位置寄存器	Position[8]	Position[9]	Position[10]	Position[11]
端口	P2.0	P2.1	P2.2	P2.3
上极限	120	120	162	162
中间态	104	104	146	146
下极限	88	88	130	130

9.2 速度与加速度解析

侧身动作是双足机器人行走的必要动作。由于双足机器人要求重心始终落在脚面边缘以内，而且行走过程中需要交替换腿，所以只有侧身动作才能将重心来回调整至 2 个脚的边缘以里。中间换脚过程存在一个过渡区域，双脚挨得近，区域较小；反之，双脚离得远，区域较大，具体计算在后面的章节讲述。

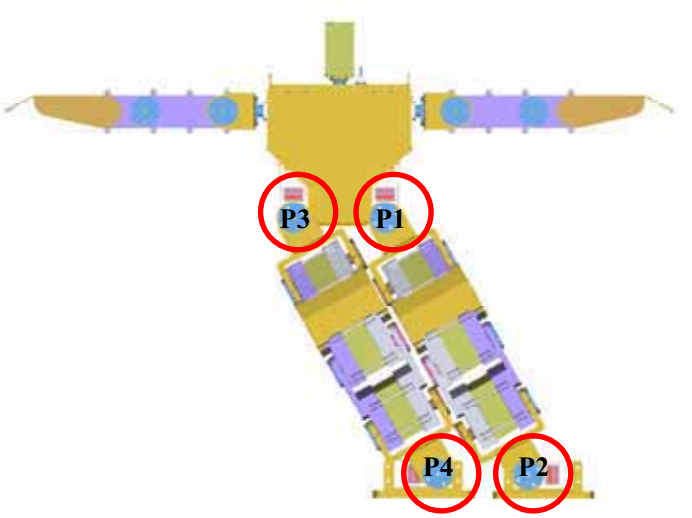
侧身过程，是为后面紧接的踏步动作做准备。那么，就存在一个衔接的问题。系统处于稳定状态，在机械上表现为机器人平稳动作即不抖动。那么，根据牛顿定律，机器人加速度为“0”，不受外力或称合力为“0”。那么，机器人运动至左、右两侧最大角度时，速度为“0”，即瞬间停止。根据左、右侧身动作的整体对称性与周期性，我们可以归纳出，中间位置时的速度最大，两端最小或为“0”，属于余弦函数或周期性循环的 2 个指数函数。

机器人的侧身要求：①侧身后双脚站立，不倾倒、不倾斜、不抖动；

②侧身后单脚站立，不倾倒、不倾斜、不抖动；

显然，单脚站立要比双脚站立难于实现。

影响机器人站立稳定性的因素主要有：
① 每个关节舵机的自抖动；
②（身+手+头）质量总和与双腿质量的比值；
③（身+头）与双手的质量比；
④ 头的质量分配；
⑤ 身的质量分配



引入系统速度平均值概念。由于系统时刻处于变速运动过程，其速度存在最大值与最小值，并在其间过渡。我们控制其最大与最小值即可控制整个系统的平均速度。

那么当 Sa 最大时，速度最小。

向左侧身第 1 族子程序中，Sa=1/2*Ea+psax；其中 psax 内存放速度平均值修正系数。

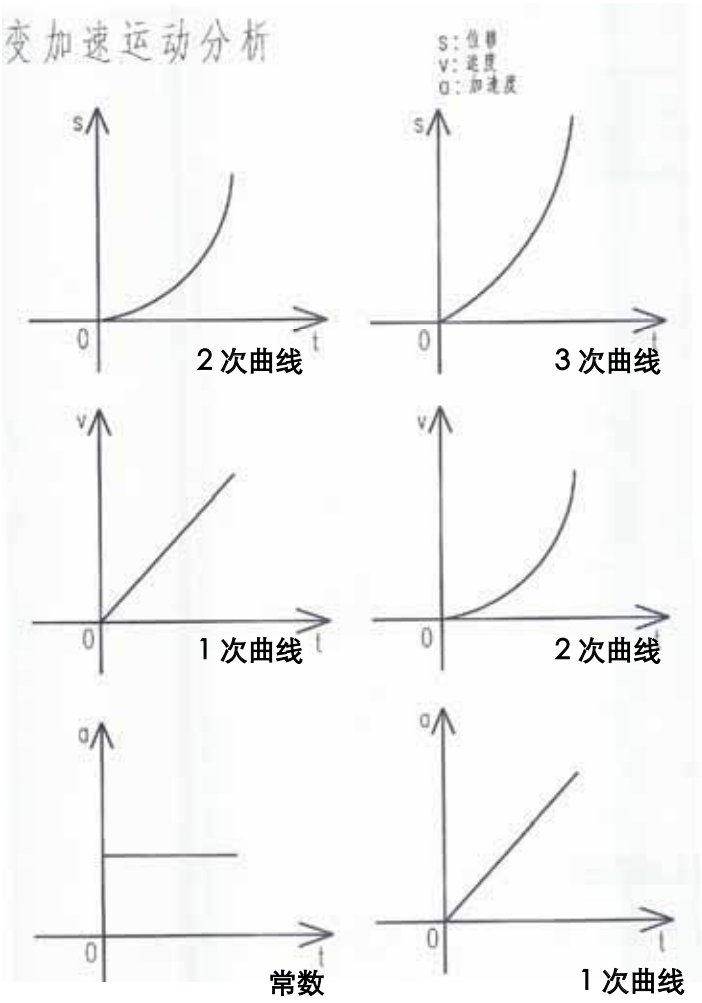
程序调用时可以预先放入初始值；

一般的，侧身运动的速度不能太快，所以取值范围 26—48。

10. 变速及变加速运动解析

10.1 变加速运动定义

所谓的变加速运动，是在匀加速运动的基础上将加速度改变的运动。



匀加速 **变加速**

图中可以看出，匀加速运动过程中，加速度为常数。这样就会给系统带来长时间的最终外力，系统会不稳定。

如果需要系统稳定，可以构造一个变加速系统：起初加速度为“0”，加速度逐渐变大，达到最大值后再逐渐减小至“0”，此时，系统停止运动。在整个过程中，加速度由生至灭，逐步过渡，属于开口向下的抛物线。

我们将加速度 a 积分，可以得到速度 v 函数；再将速度 v 积分可以得到位移 s 函数；

10.2 扫尾系数

程序调速过程中，调用 PWM_8 程序后，依靠变化的延时程序拖延下次位置数据发送的时间，以此进行调速控制。

令扫尾时间系数为 Sa

则：5ms ≤扫尾时间≤ 25ms ；△扫尾时间=20ms；

若：以 0.5 ms 为 1 个时间单位，

则：△Sa= 20ms÷0.5ms = 40

则：1≤△Sa≤40

在进行系统调速控制时，改变系统速度有 2 个方法：

- 舵机控制分辨率不变，改变下一个目标位置的发出时间，使 Sa 变化；
- Sa 不变（即每个控制周期均保持 20ms 左右），改变系统的分辨率；

由于好的数控系统，忌讳变更分辨率。我们每时每刻希望得到系统的最高分辨率，所以我们采用改变 Sa 数的方法，控制舵机速度。

扫尾系数 Sa 十分重要，在后面的速度与加速度的控制算法中会针对每种运动方式具体给出 Sa 的函数，而这些也是微分元函数表达式的重要组成部分。

10.3 积分步数系数

数控系统大量采用积分程序，那么系统进行计算时采用连续的积分运算，所以系统的积分步数很重要，根据积分步数可以时刻计算出具体的舵机位置。

令：积分步数为 Ea

则：Ea 是正数、整数，并根据具体的动作有不同的界。

一般的，腿部运动过程中 Z 平面，1≤△Ea≤76；

11. 基础动作子程序解析

11.1 P1 口 8 路单周期子程序

```
//=====================================================================
// 函数名称：void PWM_8p1( )
// 程序入口：position[0]-- position[7].
// 功能描述：在 1 个标准周期内，将 P1 口 8 个端的位置数据转换成 PWM 信号输送出去；
// PWM 信号：0.5ms—2.5ms；最短扫尾：5ms；
//=====================================================================
void PWM_8p1( )
{           uchar i=0,j=0;

          for(i=0;i<=7;i++)                    //给排序数组赋值
          {
                    paixu_ncha[i]=position[i];
          }

          sorting( );                          //调用排序函数
          N_value( );                          //调用 N 差函数
          P1=0xff;                             //使口 P1 全部拉高

           delay_500us();                      //调用延时 490us 函数
          for(i=0;i<8;i++)                     //延时输出到口 P1（八路）
          {
                    for(j=0;j<paixu_ncha[7-i];j++)
                    {
                              delay_8us();
                    }
                    P1=P1&kouchu[7-i];
          }
}
```

11.2 P2 口 8 路单周期子程序

```
//====================================================================
// 函数名称：void PWM_8p2( )
// 程序入口：position[8]-- position[15].
// 功能描述：在 1 个标准周期内，将 P2 口 8 个端的位置数据转换成 PWM 信号输送出去；
// PWM 信号：0.5ms—2.5ms；最短扫尾：5ms；
//====================================================================
void PWM_8p2( )
{       uchar i=0,j=0;

        for(i=0;i<8;i++)                //给排序数组赋值
        {
                paixu_ncha[i]=position[i+8];
        }
        sorting( );                     //调用排序函数
        N_value( );                     //调用 N 差函数
        P2=0xff;                        //使口 P2 全部拉高

        delay_500us();                  //调用延时 490us 函数
        for(i=0;i<8;i++)                //延时输出到口 P2（八路）
        {
                for(j=0;j<paixu_ncha[7-i];j++)
                {
                        delay_8us();
                }
                P2=P2&kouchu[7-i];
        }
}
```

11.3 单目标位置输出方法

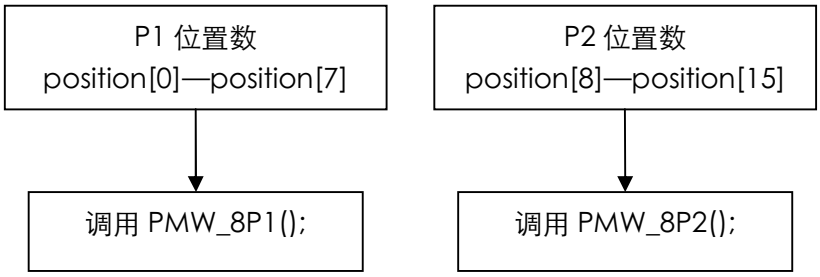
主要依靠 PWM_8P1()和 PWM_8P2()子程序实现，方法如下：

将目标位置数放入 position[0]-- position[7]寄存器内。 然后调用 PWM_8P1()子程序，即可输出 P1 口的目标位置。

同理，将目标位置数放入 position[8]-- position[15]寄存器内，然后调用 PWM_8P2()子程序，即可输出 P2 口的目标位置。

若想同时输出 P1 口和 P2 口的信号，可以依次执行，也可并行执行。

依次执行 **时间间隙长：** 放数 position[0]-- position[7] PWM_8P1(); 放数 position[8]-- position[15] PWM_8P2();	**并行** **时间间隙短** 放数 position[0]-- position[7] 放数 position[8]-- position[15] PWM_8P1(); PWM_8P2();	1 个周期一般为 3 ms -20ms，2 个端口依次执行，要考虑周期时间对扫尾时间的影响，即影响调速。



12. 语言程序架构解析

12.1 C 语言嵌入汇编

由于机器人要求一个很准的时钟，并通过这个时钟做准确的延时，所以我们需要使用 51 的汇编语言帮助。但是在一些处理不严格的地方，我们可以 C 语言的模块化编程方法帮助完成。在这里我们使用了 C 语言嵌入汇编的技术。按文件分类，我们一共有两个文件：一个是 ASM 文件，还有一个是 C 的文件。他们需要包含在一个工程文件中才能编译通过。

12.2 汇编语言被嵌入的说明

在人形机器人程序中，由于部分是使用的汇编语言编写，为了清晰和直观。我们将汇编程序单独摘出来做成 “.asm”的文件，在工程中，我们应该将文件关联到工程目录下，一旦关联，C 程序就可以调用汇编子函数了。我们采用模块化编程方法，在调用汇编程序时也是按照子函数的方式直接调用的。使用十分方便。

12.3 KEIL C 下的具体操作步骤

12.3.1 在 KEIL C 下建立工程（project）

在“project”菜单下选择“New project”，这时会有对话框询问你将工程保存在什么目录下，并询问工程名称，这个名称将使系统相关的文件都已这个名字为开头，包括我们将要用到 HEX 文件。如图所示：

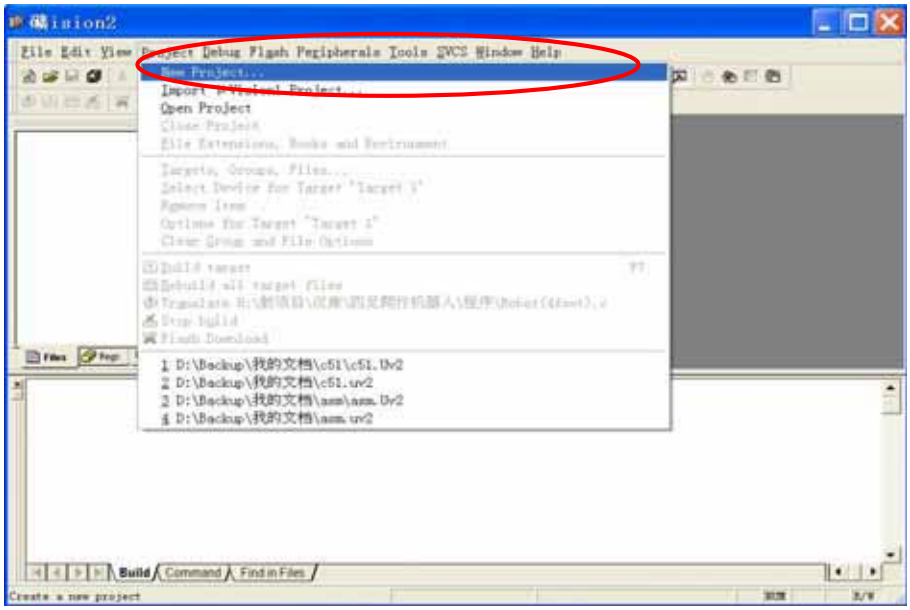


图 12.1 建立工程

12.3.2 选择开发芯片类型

在设置好文件名和目录以后，系统会询问你所使用的芯片类型。我们这里选用宏晶单片机：STC12C5410A。点击“确定”完成工程向导。

12.3.3 添加程序文件到工程

将制作好的 C 程序和汇编文件（必须以“.C”“.ASM”为后缀文件名）加入到新的工程中去。我们可以在下图中的对应位置，点击右键并左键点击对应选项添加文件。

这里需要说明一下，如果添加的文件只有 C 程序而没有添加汇编文件。则编译的时候会出错。

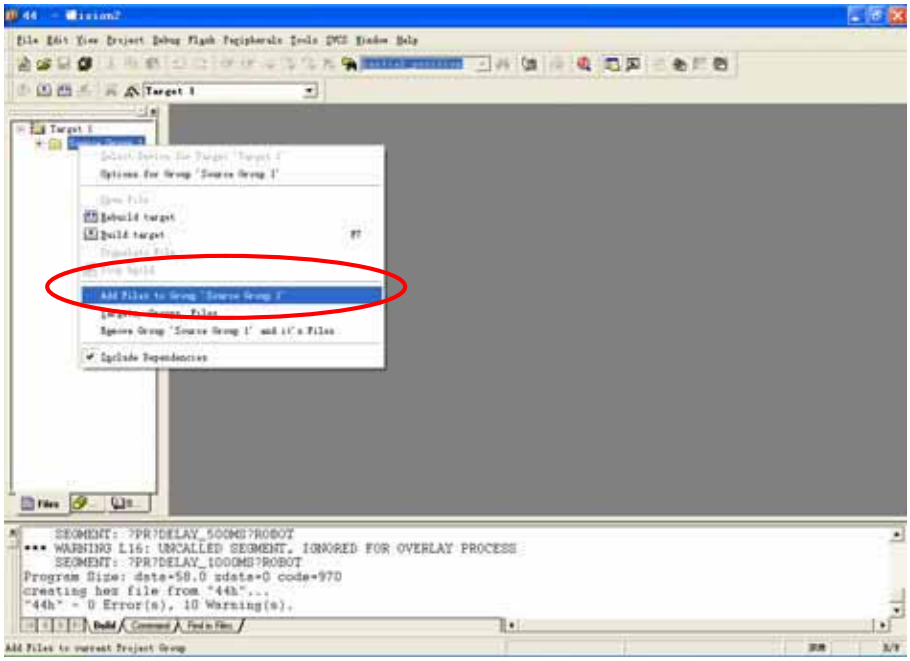


图 12.2 在工程中添加文件

12.3.4 设置工程属性

每个工程的要求不一样，在 KEIL C 中为这些属性做了一个属性选项。

首先在“project”菜单下选择“options for File‘xxxxx’”。如下图所示：

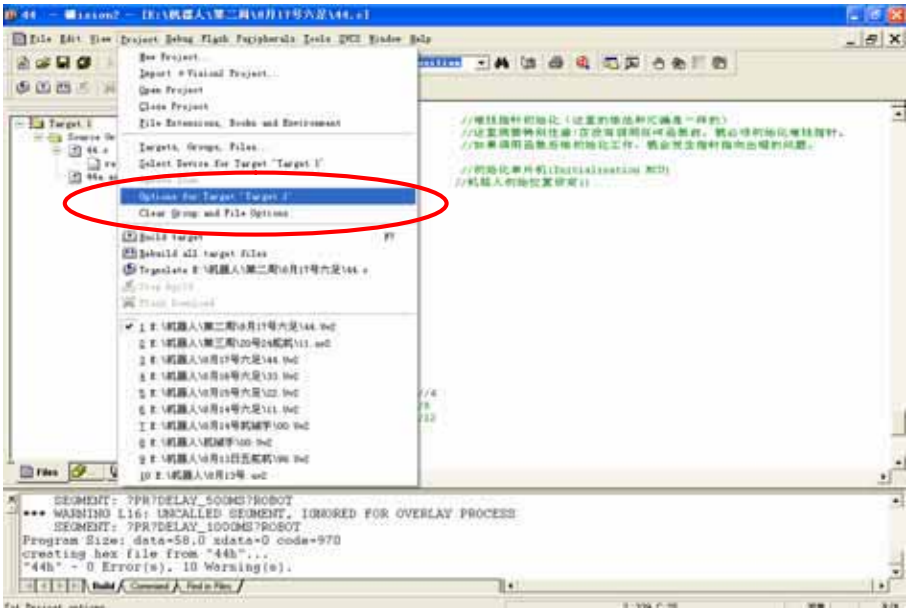


图 12.3 属性设置

之后就会弹出工程属性对话框，在第三个“Output”选项卡下，将生成 HEX 文件的选项打上对勾，如下图所示：

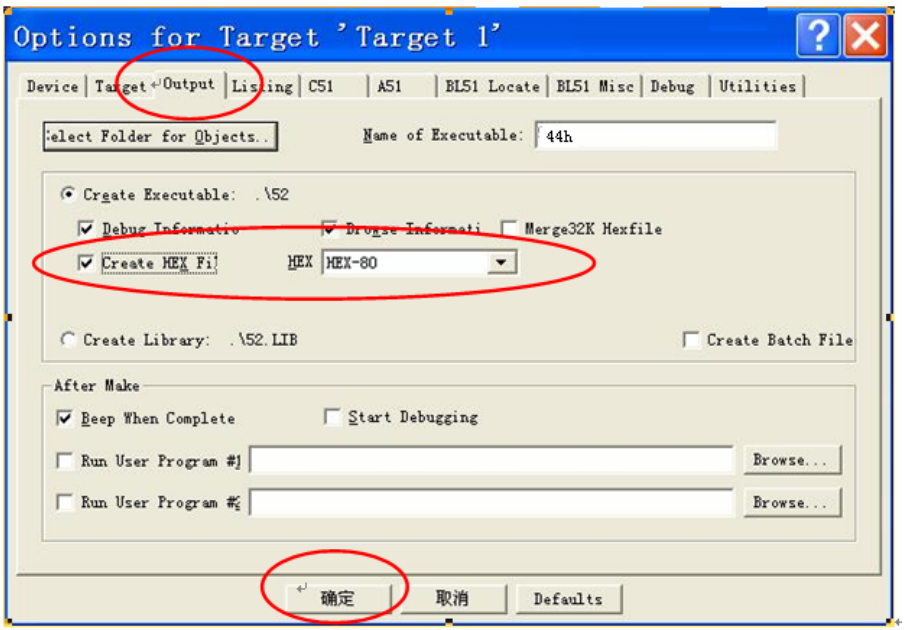


图 12.4 属性设置选项卡

点击确定完成工程属性的调整，这样我们在之后生成的过程中就会生成 HEX 文件。对于后面的烧录软件 STC_ISP 来说，烧写需要的是 HEX 文件。

我们可以选择完全编译，然后看看是否生成了 HEX 文件。如下图所示：

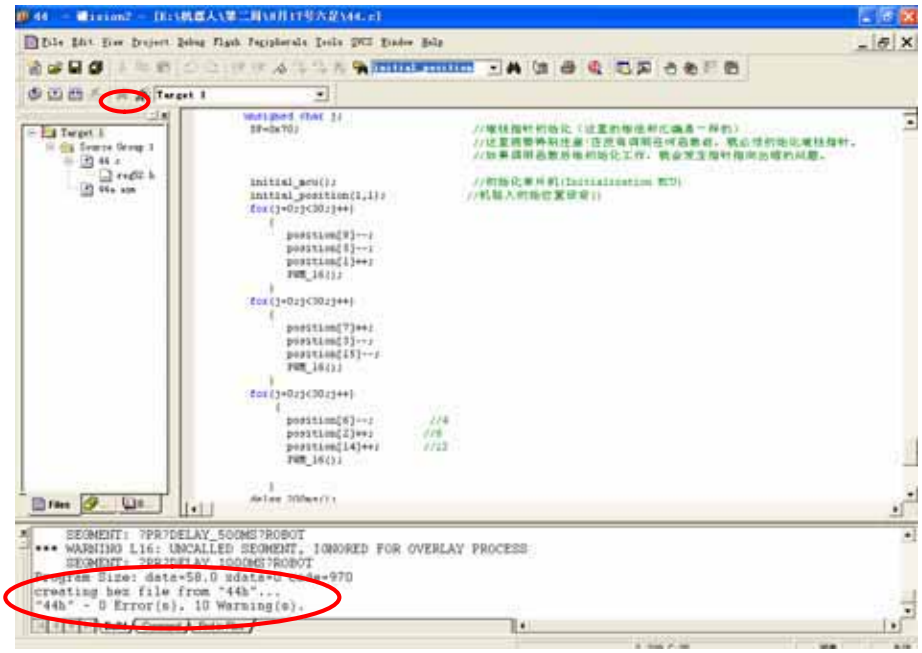


图 12.5 编译界面

这样我们就可以使用工程中的 HEX 文件通过 STC _ISP_V4.7 烧录软件烧录 STC12C5410AD 了。STC 单片机是基于 51 控制核的高速单片机。对于程序的编译和链接，我们可以使用 KEIL C 帮助完成。

12.3.5 KEIL C 的安装

图 12.6 首先找到安装文件目录：“Keil 7.2”

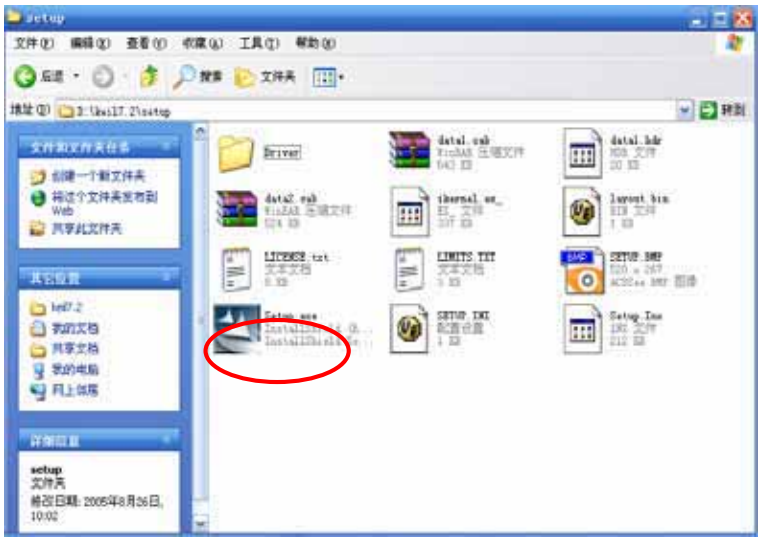


图 12.7 双击“setup”文件

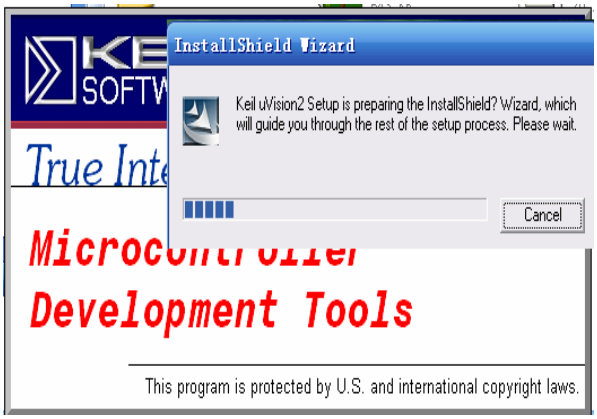


图 12.8 安装进程

图 12.9 点击“Full Version”

接下来点击“NEXT”、“YES”直到出现以下界面：

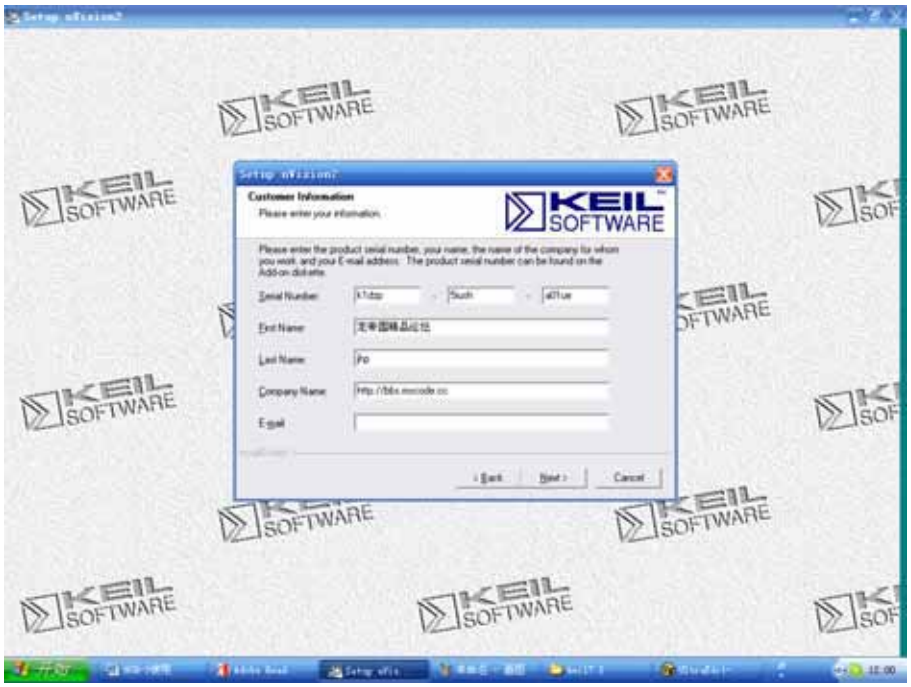


图 12.10 从安装文件夹下找到“序列号”填入

接下来点击“NEXT”、直到“FINISH”结束安装。

13、程序下载软件说明

在对芯片进行编程时，我们使用 STC 公司提供的烧录软件 STC_ISP。

STC_ISP_V4.7 是宏晶科技公司提供的能直接在编程者电脑上使用的 ISP 在线下载方式，将用户程序目标代码(.hex 文件)，下载进 STC 单片机的软件。非安装版,自解压 直接执行 STC-ISP.exe 即可 。

它可以通过普通的串口（COM）烧写单片机的程序。软件运行稳定。操作相对方便。

软件的操作界面如下：

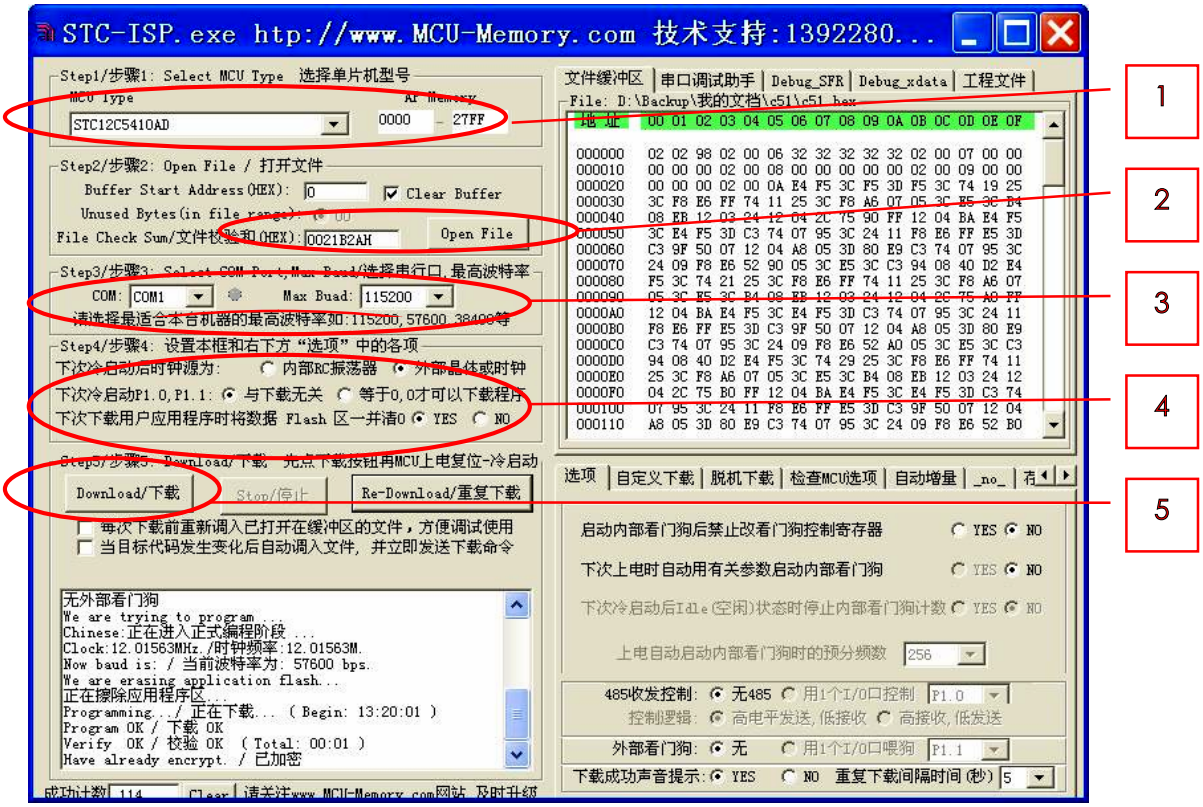


图 13.1 STC_ISP 软件界面

下面给出操作的具体步骤：

1. 选择芯片类型；
2. 选择要烧写的 HEX 文件；
3. 设置串行端口和波特率，这里要注意端口号，波特率的选择比较任意一般为 38400；
4. 选择外部晶振；与下载无关；清 FLASH 区；
5. 点击下载按钮后，打开单片机供电电源。在上电后单片机会自动进入编程状态。通过提示可以判断是否下载完成。

注意：

1. **在选择芯片时一定要看清型号，很容易选错；**
2. **应选择外部晶振，如果选择内部晶振，单片可以工作，但是舵机的控制会出现问题；**
3. **一定要先点编程按钮，然后再打开电源，否则下载不能成功。**

14. 应用方案

人形机器人一直是机器人领域研究的热点,它集中了机械工程、电子工程、计算机工程、信息工程、自动控制工程以及人工智能和仿生学等多种学科的最新科研成果,代表了机电一体化的最高成就,是目前科技发展最活跃的领域之一。

人形机器人为高校师生提供了一个综合的机器人教育和研究平台，在这个新颖有趣的平台上，可以引导同学进行单片机编程、数字/模拟电子、传感器应用与检测、机器人控制等课程的学习，加强同学们理论联系实践、提高创新动手能力，提升就业竞争力。

适用场合：

- 工程训练中心的科技创新训练；
- 电气自动化、自动控制、电子信息、计算机科学等相关专业的教学和课外实践；
- 各类机器人相关的比赛：舞蹈机器人、足球机器人等；

课程安排：

（1）**初级阶段**—以机器人底层控制为主

学习内容：机器人机械结构认知组装、单片机编程控制、人形机器人控制算法学习。

完成项目：机器人单脚站立、走路、跑步、前滚翻、后滚翻、侧翻、双手及单手俯卧撑、倒立；多个机器人进行舞蹈表演。

（2）**高级阶段**—重点在机器人扩展和算法

学习内容：机器人结构研究、机器人步伐函数研究、机器人运动函数（基于微分方程）研究；扩展摄像头、无线模块、陀螺仪、角度传感器、测距/避障传感器等；扩展 ARM、FPGA、DSP 高端处理器；

完成项目：机器人平衡控制研究；带图像视觉的足球机器人设计；仿人智能机器人设计等；

关于我们

公司简介

亿学通™ 电子是纳科斯科技旗下的高校事业部，成立于 2005 年底。专精于高校竞赛创新产品和设备的开发、生产及销售。和全国 1400 余所高校有着广泛的联系和合作。

亿学通™ 坚持“合适即最好”的经营理念，倡导“创新实践、学以致用”的教育理念，以市场需求为导向、以科技创新为依托，为客户研发最有针对性的产品和设备，让更多的同学能在亿学通™ 平台上得到锻炼和能力的提升是公司的服务宗旨。

现在产品种类已发展成“创新教学产品、机器人/智能车、电子实习套件、电子竞赛套件、配套电子模块、传感器模块等六大类近两百种产品。

公司具有专业的研发团队和第一流市场调研人员的。可以保证我们能够为客户提供最合适的产品。

远景规划

让一亿大学生在**亿学通™** 创新平台中受益。

经营理念

合适即最好！Be Appropriate to be Best!

注：为高校创新实践量身定制最合适的实践创新教育产品。

经营策略

易：寓教于乐、轻松掌握；

新：技术新潮、形式新颖；

精：树精品意识、创优秀品牌；

专：以专业的眼光做产品；以专一的态度做事情

联系方式

电话：**010-62669059**

邮箱：**61mcu@61mcu.com**

网址：**http://www.61mcu.com**

地址：北京市海淀区上地七街国际创业园 2 号院 C 座 812

账户信息

中国工商银行对公账户 (对公业务窗口办理,汇率:1%,最高不超过 50 元,2-3 个工作日可以到帐)

户名：北京纳克斯科技发展有限公司

开户行：中国工商银行海淀支行营业部

帐号：**0200 0496 0920 0322 217**

汇款备注

当您汇款后，请将汇款底联传真或扫描后 E-mail 给我们；我们确认您的汇款后，会在约定时间内为您发货；

如果您方便使用个人汇款，请您到公司网站的“付款方式”栏查看，或发 **mail** 到 **sale@61mcu.com** 索取个人汇款信息