机器甲虫的设计综合（技术揭密）

第一章 甲虫的电子设计

1.1 设计构想

我们是一批制作机器生物的爱好者，致力于机器生物的教育和普及工作。机器甲虫是我们设计的第一个机器生物，从仿生学的角度，昆虫的生理构造和行为是比较容易模仿的。即使如此，昆虫的生理构造和行为为还是相当复杂，如苍蝇的复眼、蚂蚁群的社会化行为等等。因此我们在以下的设计中遵循去繁从简的原则，让大家体会初步设计机器生物的精髓所在。

高级生物人类的认识过程为：感性认识→理性认识→实践；甲虫对外界的反应过程：感知变化→经验判断→反应。两者的区别在于人类的认识会有感性认识到理性认识、理性认识到实践的飞跃，而甲虫始终停留在感性认识（感知、经验）的低层次。我们设计的机器甲虫就相应地拥有三大功能模块：感觉模块、判断模块、反应模块。连接着三大模块的是电子线路和电信号，对应于甲虫的神经连接系统和神经信号。

1.2 相关知识

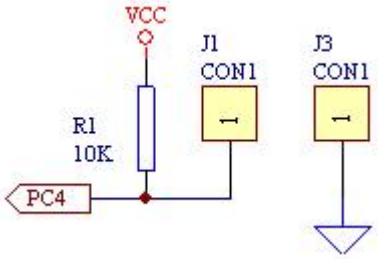
内容正在完善中.....

1.3 感觉模块

感觉模块的功能是感知外界的声、光、触碰等等变化，转化成电信号，通知判断模块。感觉有清晰和模糊之分，这是质和量的辩证关系，例如有无光和光的强弱就是一对清晰与模糊的关系。感觉模块产生的电信号亦有数字和模拟之分，数字信号只有两种状态就是 0 和 1，模拟信号是连续变化的量。

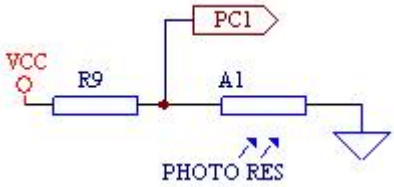
1.3.1 触觉

对于简单触觉，有碰到和没有碰到两种状态，我们可以采用数字信号来描述。为了保证触觉的全方位，对于甲虫设计了三个相同的模块（如下图）作为左、右触角和尾巴。R1 为上拉电阻，J1 连接触碰圈，J3 连接触碰杆，PC4 为输出的电信号。假如 J1 和 J3 碰在一起， PC4 输出 0；J1 和 J3 没有相碰，则 PC4 输出 1。



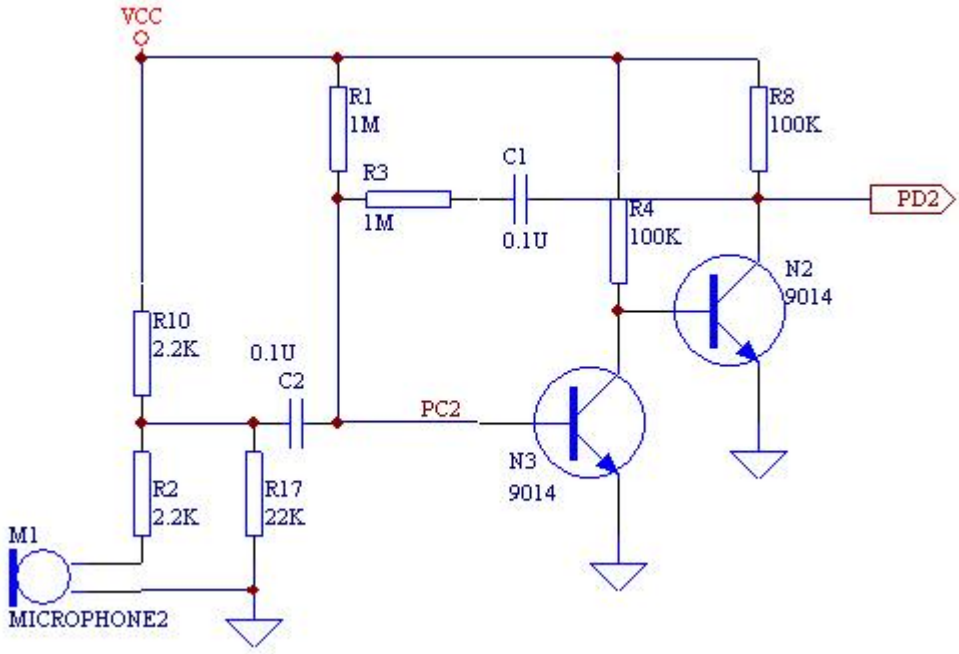
1.3.2 视觉

视觉是一种非常复杂的感觉，有两种特性——色彩和明度。色彩由红绿蓝三原色比例产生，而明度是三原色各自的数量。这里我们只关心最简单的环境光综合明度（就是光的强弱），对光的色彩、图像等复杂视觉不做处理。电路设计如下图。R9 是分压电阻，A1 是光敏电阻，PC1 为输出的电信号。光敏电阻的阻值随光线的增强而变小，PC1 输出的电压值随之降低。PC1 输出的电压值是模拟量。



1.3.3 听觉

听觉也非常复杂，其特征有音调、节拍、音色、声强等。音调是声音的频率，节拍是指音与音之间的长短，音色是声音的振动特性，声强是声音的大小。我们的设计只与声强和简单节拍有关，这样会比较容易实现。麦克风是我们最常见的声电转换产品，咪头是麦克风的核心器件。我们的电路会围绕咪头进行设计。如下图，R2、R10、R17 为分压电阻，用来选择检测范围，C2 为耦合电容。通电后 C1 被充满电。无声音信号时，延时控制电路中两只晶体管 N3 导通、N2 截止，PD2 输出高电平。当咪头 M1 接收到声音信号时，输出脉冲经 C2 耦合至延时控制电路，N3 截止、N2 导通，输出由高电平变为低电平。声音信号消失后，延时控制电路输出仍维持数十毫秒的低电平。直至 C1 被充满，PD2 输出高电平。这里我们用电路实现了模拟量向数字量的转换。



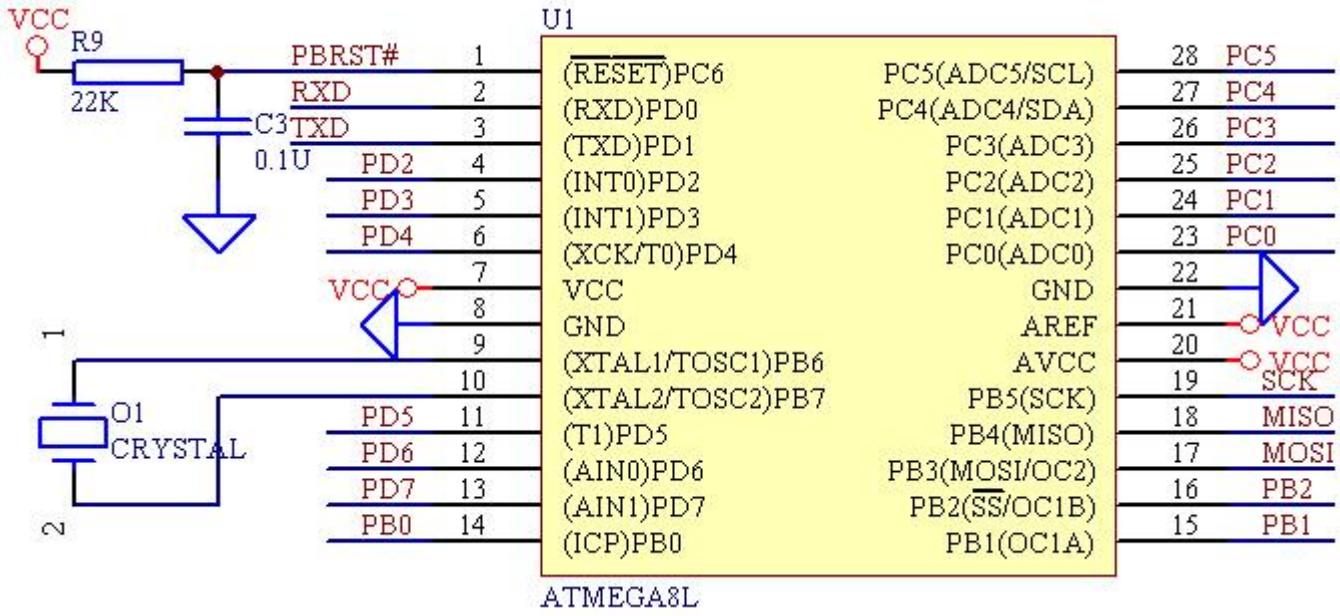
以上是我们对感觉模块的设计，几乎都是最简单和最基本的。真实世界里，感觉是丰富多彩的，如味觉、嗅觉、温度、湿度、压力等等，每一种感觉都有其独特的特性，我们要把它们用电子的方式描述出来，是庞大艰辛而又非常有趣的工作。

1.4 判断模块

判断模块的功能是接收来自感觉模块的信息，做经验判断，发出反应信息。

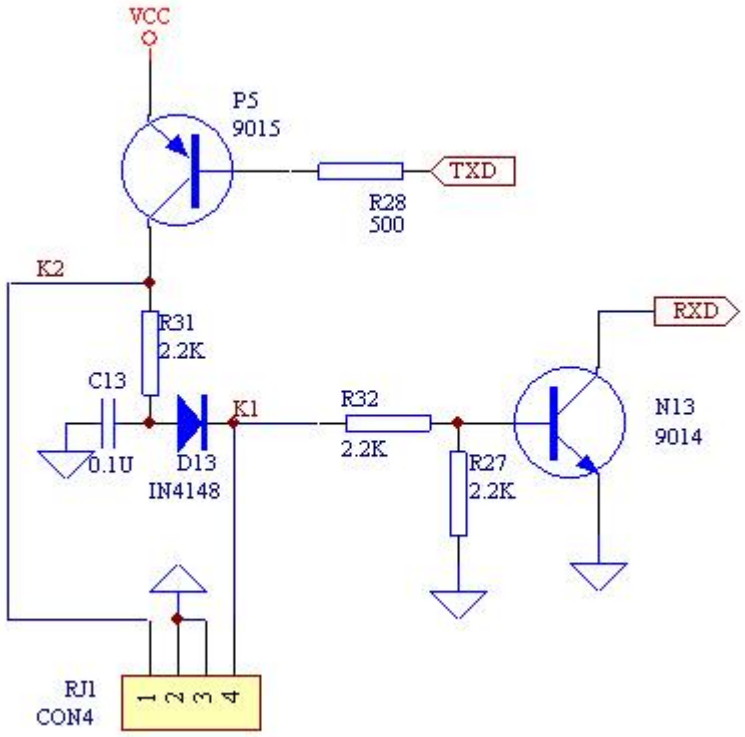
1.4.1 大脑

来自感觉模块的信息有模拟的也有数字的，我们又希望机器甲虫拥有人工智能，最简单的选择无过单片机。我们选择了 ATEML 的 ATMEGA8L 芯片。如下图，它有六路A/D 器(模拟/数字转换器)，ADC0~ADC5，与双向I/O 口 PC0~PC5 使用相同管脚，另有双向 I/O 口 PD2~PD7、PB0~PB5 可用作数字电信号的接收或输出。O1 是晶振，相当于单片机的“心脏”，单片机的工作时钟就是靠它和单片机内部的振荡放大器产生。R9、C3 组成上电延时电路，使单片机在通电时可靠复位正常工作。RXD、TXD 是分别是单片机的串行通讯接收和发送端口。



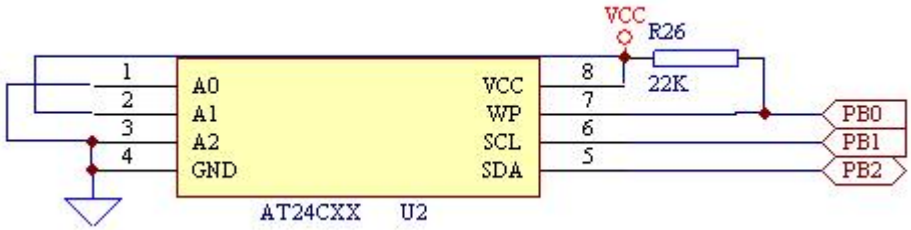
1.4.2 通讯

由于计算机的人机对话界面比较友好，我们可以在计算机中把人工智能的参数设置好，再通过通讯端口把这些参数传输给机器甲虫。单片机的程序就是根据这些参数以及感觉模块的信息做出判断，发出相应的电信号到反应模块。单片机程序设计等相关内容我们在“下一章单片机程序介绍 1.3”会详细介绍。计算机中的串行口即 RS232 口的接口电平标准，不同于单片机的端口电平，因此需要作 TTL 电平转换。如下图，可实现半全双工串行通讯。TXD 为本端发送输入，RXD 为本端接收输出，K1 为计算机端发送输入，K2 为计算机端接收输出。首先看发送过程，TXD 为高电平时，P5 截止，K1 为低电平，通过R31、D13 使 K2 亦为低电平；TXD 为低电平时，P5 导通，使K2 为高电平。再看接收过程，K1 为高电平时，经过 N13 反相，RXD 输出低电平；K1 为低电平时，N13 截止，RXD（通过单片机内部管脚上拉电阻）输出高电平。注意到计算机串口的高电平为10V，低电平为-10V，加接 R27 用于保护 N13 不被反向击穿，R27 还可以保证在不通讯时，N13 保持截止状态。



1.4.3 记忆

计算机传输过来的参数要保存起来，即使完全掉电仍可记忆，就需要用到 EEPROM 之类的非易失性存储器。此电路适用 AT24C02、AT24C04、AT24C08 以及 AT24C16 等二线传输 EEPROM，容量分别对应于 2K、4K、8K、16K 位。高位地址线 A0、A1、A2 设置为二进制 010，对于 AT24C16 来讲地址设置无效。PB1 为同步时钟，PB2 为数据输入输出口，电路上电时由于 PB1、PB2 的状态不可知，有可能改变 EEPROM 的数据，R26 为保护上拉电阻，可保证 EEPROM 数据不容易被意外更改。需要改变 EEPROM 数据时，要先把 PB0 置低。



我们现在设计的判断模块，将会非常大程度地受到人为因素的影响，进化理论对机器甲虫无效。也许将来我们可以设计出符合进化理论的机器甲虫来，拥有进化功能，让它和自然界的甲虫来一下优胜劣汰。

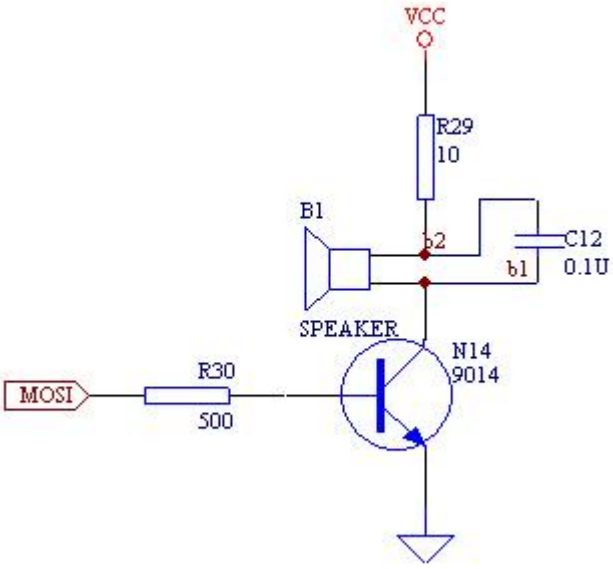
1.5 反应模块

反应模块的功能是接收来自判断模块的信息，做出各种不同的声、光、动作反应。

1.5.1 发声

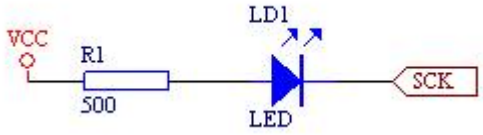
下面介绍喇叭电路的设计。MOSI 为声音频率节拍输入，B1 为喇叭，R29 为限流电阻，R30 为基极电阻。NPN

三级管 N14 用作开关管，电容 C5 吸收电路噪声、延长喇叭寿命。喇叭电路可以播放单一音色的 MIDI 格式音乐。



1.5.2 发光

下面介绍发光电路的设计。R1 为限流电阻，LD1 为发光二极管，SCK 为控制端口。SCK 为低电平时，LD1 有电流流过，发光二极管亮；SCK 为高电平时，LD1 没有电流流过，发光二极管灭。

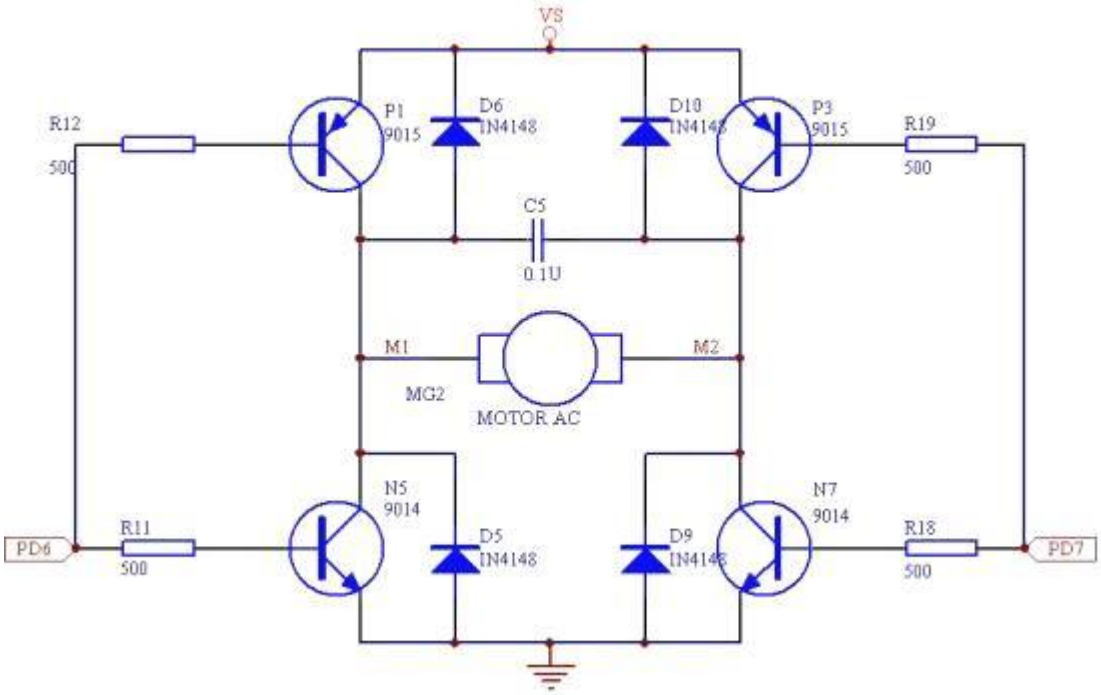


1.5.3 动作

下面介绍微型马达（电机）驱动电路的设计。微型马达的工作电压为 3.6v，工作电流 120mA 以上，要求双向可控制驱动，以实现前进、后退及停止。采用如下图桥式驱动电路，三极管全部起开关作用，基极电阻选用 500 欧，PD6、PD7 为信号输入端，状态控制如下表：

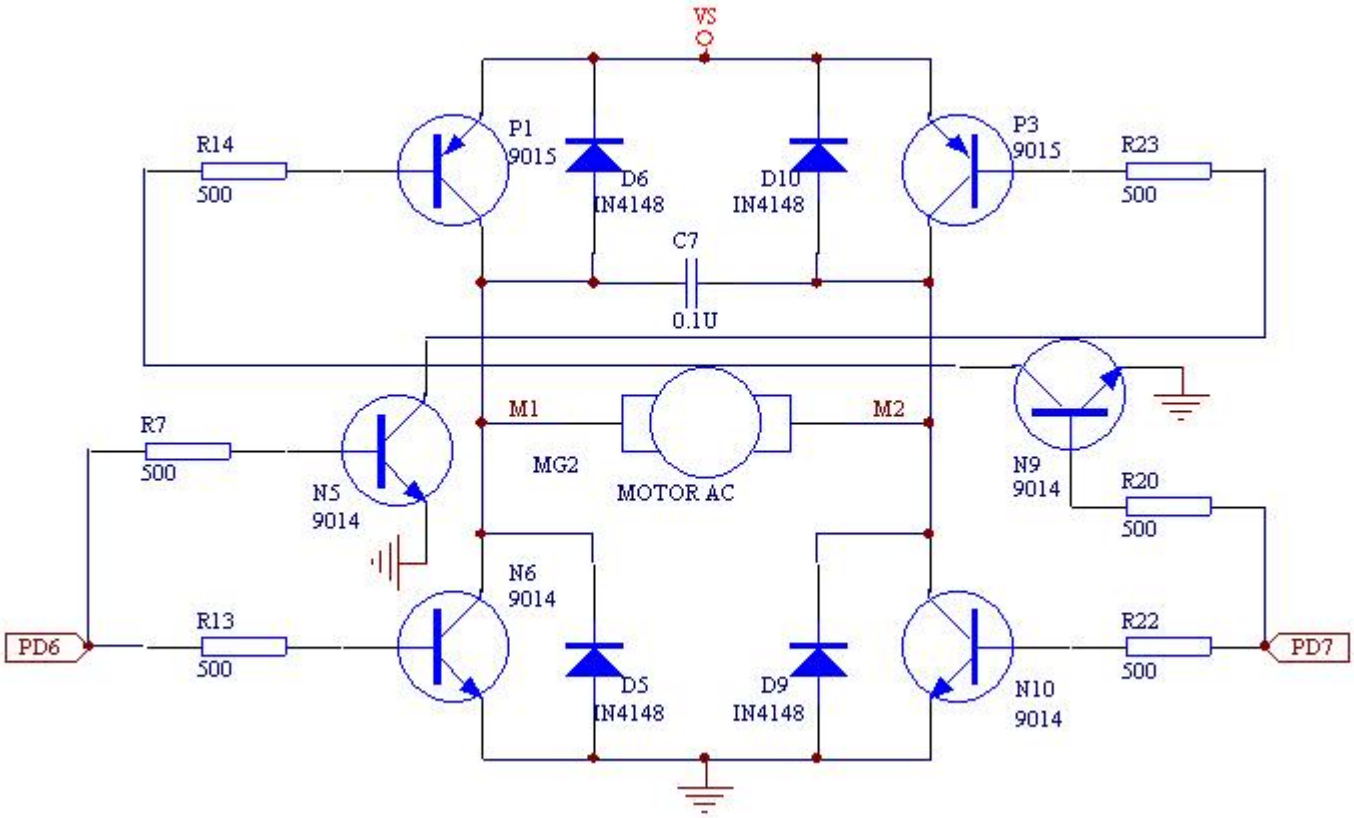
PD6	PD7	P1	P3	N5	N7	电流方向
0	0	开通	开通	截止	截止	无
0	1	开通	截止	截止	开通	M1=>M2
1	0	截止	开通	开通	截止	M2=>M1
1	1	截止	截止	开通	开通	无
Z	Z	开通	开通	开通	开通	P1=>N5、P3=>N7

注：Z 为高阻态。二极管 D5、D6、D9、D10 起内部电路保护作用，释放微型马达被机械转动时产生的电。电容 C5 吸收电路噪声、平衡左右电路器件差异、防止微型马达被干扰转动，考虑到电路的轴对称性，应选用无极性电容。



考虑低功耗要求，即控制输入脚高阻态时，电路进入低功耗。上面的设计有缺陷，它在输入高阻态时，P1=>N5 以及 P3=>N7 会有漏电电流（此时马达两端电压相同亦不会转动）改进的电路如下图，状态控制如下表：（试比较与上表有何不同）

PD6	PD7	P1	P3	N5	N6	N9	N10	电流方向
0	0	截止	截止	截止	截止	截止	截止	无
0	1	开通	截止	截止	截止	开通	开通	M1=>M2
1	0	截止	开通	开通	开通	截止	截止	M2=>M1
1	1	开通	开通	开通	开通	开通	开通	P1=>N6、P3=>N10
Z	Z	截止	截止	截止	截止	截止	截止	无



第二章 单片机程序设计

2.1 你准备好了吗

假如你已经看完<<甲虫的电子设计>>这一章，相信你对心爱的甲虫已经有一定的了解，在确定甲虫的个性的时候将会有事半功倍之效；假如你对电子设计不大感兴趣，那也没关系，只要你认真学习这一章，同样能够使甲虫听你的指挥，确定它的个性。

在确定你心爱的甲虫的个性之前，您是否已经熟悉C 语言的编程呢？假如还没有，那你得赶快补一补，不然，甲虫将会不听你的话。

2.2 了解你的 IQBug

机器甲虫拥有三大功能模块：感觉模块、判断模块、反应模块。感觉模块的功能是感知外界的声、光、触碰等等变化，转化成电信号，通知判断模块。判断模块的功能是接收来自感觉模块的信息，做经验判断，发出反应信息。反应模块的功能是接收来自判断模块的信息，做出各种不同的声、光、动作反应。

遵循模块化的设计思路,机器甲虫的各种功能都已封装成函数， 只要调用函数就可实现相应的功能，下面列出了各个功能函数及其所在文件。

系统初始化函数 init_devices	封装在文件 init.h
延时函数 delay	封装在文件 init.h

电压检测函数 check_power　　封装在文件 power.h

串行通信

接受串行数据函数 receive　　封装在文件 uart.h

发送串行数据函数 send　　封装在文件 uart.h

EEPROM 读写

读数据函数 x24c08_read　　封装在文件 eeprom.h

写数据函数 x24c08_write　　封装在文件 eeprom.h

感觉模块

感光检测函数 check_photo　　封装在文件 photo.h

声音检测函数 check_mic　　封装在文件 mic.h

触碰检测函数 check_touch　　封装在文件 touch.h

反应模块

发光控制函数 led_process　　封装在文件 led.h

发声控制函数 speak_process　　封装在文件 speak.h

动作控制函数 motor_process　　封装在文件 motor.h

2.3 教教你的 IQBug

在开始设计你心爱的甲虫的个性之前， 你必须先建立一个新的文件，命名为*.c，例如 main.c，然后把文件 init.h 包含进来，调用函数 init_devices（）进行初始化。

```
#include “init.h” //引入文件
Void main（void）
{
//……
init_devices（）；//系统进行初始化
//……
}
```

由于 delay 函数在后面经常使用，故在此先作介绍。

延时函数：void delay(unsigned int ms);

功能:延时 ms 毫秒

参数含义：

ms：延时时间，单位毫秒

2.3.1 教你的 IQBug“眨眼”

要使你心爱的甲虫会“眨眼”,只需调用 led_process 这个函数，下面我们看看函数 led_process 的详细定义。

Void led_process(unsigned char which_eye, unsigned char state);

功能:根据参数对相应发光眼睛进行处理

参数含义:

which_eye,表示哪一个发光眼睛

which_eye=0,表示左发光眼

which_eye=1,表示右发光眼

state,表示相应发光眼睛状态

state=0,表示不亮

state=1,表示亮

例子：左发光眼亮 1 秒，右发光眼亮 1 秒

```
#include “init.h”
void main(void)
{
init_devices();
led_process(0,1); //右发光眼不亮
delay(1000); //延时 1000 毫秒，即 1 秒
led_process(0,0); //左发光不亮
led_process(1,1); //右发光眼亮
delay(1000); //延时 1000 毫秒，即 1 秒
}
```

2.3.2 教你的 IQBug“走路”

要使你心爱的甲虫会“走路”,只需调用 motor_process 这个函数，下面看看函数 motor_process 的详细定义。

void motor_process（unsigned char which_motor,unsigned char state）;

功能:根据参数对相应小腿进行处理

参数含义:

which_motor,表示哪一个小腿

which_motor=0,表示左小腿

which_motor=1,表示右小腿

state,表示相应小腿状态

state=0,表示不动

state=1,表示前进

state=2,表示后退

例子:前进 3 秒,停 2 秒,后退 1 秒

```
#include “init.h”
void main(void)
{
init_devices();
motor_process(0,1);//左小腿前进
motor_process(1,1);//右小腿前进
delay(3000);
motor_process(0,0); //左小腿停
motor_process(1,0); //右小腿停
delay(1000);
motor_process(0,2); //左小腿后退
motor_process(1,2); //右小腿后退
delay(2000);
}
```

2.3.3 教你的 IQBug“发声”

要使你心爱的甲虫会“发声”,只需调用 speak_process 这个函数，下面看看函数 speak_process 的详细定义。

void speak_process（unsigned int freq）；

功能:发出频率为 freq HZ 的音

参数含义:

freq,音的频率

例子 1:让甲虫发出几个不同频率的声音

```
#include “init.h”
void main(void)
{
init_devices();
timer1_init(); //初始化定时器 1
//(在发声之前,必须先初始化定时器 1,否则不能发声)
speak_process(262);//发出频率为 262 HZ 的音
delay(1000);
speak_process(294); //发出频率为 294 HZ 的音
delay(1000);
speak_process(330); //发出频率为 330 HZ 的音
delay(1000);
timer1_stop();//关闭定时器 1
//(在发声结束后,必须关闭定时器 1,否则会不断地发出最后一个音)
}
```

或许你已经注意到,上面的例子,甲虫发出的音相当于简谱的音阶“1”、“2”、“3”，要使甲虫发出更悦耳动听的声

音，就得先弄清简谱的音阶与频率的关系。

例子 2：唱出音阶“1”、“2”、“3”、“4”、“5”、“6”、“7”

```
#include “init.h”
void main(void)
{
init_devices();
timer1_init();
speak_process(262); //相当于音阶的“1”
delay(1000);
speak_process(294); //相当于音阶的“2”
delay(1000);
speak_process(330); //相当于音阶的“3”
delay(1000);
speak_process(350); //相当于音阶的“4”
delay(1000);
speak_process(393); //相当于音阶的“5”
delay(1000);
speak_process(441); //相当于音阶的“6”
delay(1000);
speak_process(495); //相当于音阶的“7”
delay(1000);
timer1_stop();
}
```

2.3.4 教你的 IQBug“躲避强光”

要使你心爱的甲虫能够“躲避强光”,先调用 check_photo 这个函数，检测当前光线情况，然后根据这个情况再作相应的处理。

下面我们看看函数 check_photo 的详细定义。

unsigned char check_photo(unsigned char which_eye);

功能:根据参数检测相应感光眼睛光线情况

参数含义:

which_eye,表示哪一个感光眼睛

which_eye=0,表示右感光眼睛

which_eye=1,表示左感光眼睛

返回值含义：表示光线强弱等级

返回值范围：0~16

值越大，光线越弱

值越小，光线越强

例子：教你的 IQBug“躲避强光”

```
#include “init.h”
void main(void)
{
unsigned char left_temp;//记录左感光眼检测到的光线情况
unsigned char right_temp;//记录右感光眼检测到的光线情况
init_devices();
while(1)
{
left_temp=check_photo(1);//左感光眼检测光线情况
right_temp=check_photo(0);//右感光眼检测光线情况
if(left_temp<7)//判断左边光线情况
//7 为光强等级中值，小于 7 表示左边光线较强，宜走到右边去
{
motor_process(0,2); //先后腿，是为了避免前边有障外物
motor_process(1,2);
delay(500);
motor_process(0,1); //左脚前进，右脚不动，则右转
motor_process(1,0);
delay(100); //要转某一角度，需要调整这个时间
// motor_process(0,0); //或者左脚不动，右脚后腿，也可右转
// motor_process(1,2);
// delay(100);
}
if(right_temp<7) //判断右边光线情况
//7 为光强等级中值，小于 7 表示右边光线较强，宜走到左边去
{
motor_process(0,2);//先后腿，是为了避免前边有障外物
motor_process(1,2);
delay(500);
motor_process(0,0); //左脚不动，右脚前进，则左转
motor_process(1,1);
delay(100); //要转某一角度，需要调整这个时间
// motor_process(0,2); //或者左脚后退，右脚不动，也可左转
// motor_process(1,0);
// delay(100);
}
motor_process(0,1); //继续前进
motor_process(1,1);
```

```
delay(500);
}
}
```

2.3.5 教你的 IQBug“听你的指挥”

要使你心爱的甲虫能够“听你的指挥”,先调用 check_mic 这个函数，检测当前发生声音的次数，然后根据这个情况再作相应的处理。

下面我们看看函数 check_mic 的详细定义。

```
unsigned char check_mic(void);
```

功能:检测发生声音的次数

返回值含义:表示发生声音的次数

例子：如果听到声音，听到多少声，叫多少声；否则，不发声

```
#include “init.h”
void main(void)
{
unsigned char mic_temp;//记录甲虫检测当前发生声音的次数
unsigned char i; //定义变量
init_devices();
while(1)
{
mic_temp=check_mic(); //检测当前发生声音的次数
delay(2);
//延时 2 毫秒
//要使发挥 check_mic 的功能，必须延时
//延时时间决定甲虫听觉的灵敏度，延时越小，听觉越灵敏
for(i=0;i<mic_temp;i++)//听到多少声，叫多少声
{
timer1_init();
speak_process(262);//叫一声
delay(1000);
timer1_stop();//停一下
delay(1000);
}
}
}
```

2.3.6 教你的 IQBug“躲避碰撞”

要使你心爱的甲虫能够“躲避碰撞”,先调用 check_touch 这个函数，检测当前发生的触碰情况，然后根据这个情况再作相应的处理。

下面我们看看函数 check_touch 的详细定义。

unsigned char check_touch(void);

功能:检测触角的触碰情况

返回值含义:

0x01，表示碰到右触角

0x02，表示碰到尾巴

0x03，表示碰到右触角和尾巴

0x04，表示碰到左触角

0x05，表示碰到左触角和右触角

0x06，表示碰到左触角和尾巴

0x07，表示碰到左触角、右触角和尾巴

0x0d，表示没有发生触碰

例子:碰到左触角,后腿,右转，左发光眼亮；碰到右触角,后腿,左转，右发光眼亮；否则前进，发光眼不亮

```
#include “init.h”
void main(void)
{
unsigned char temp;//记录甲虫检测触角的触碰情况
init_devices();
while(1)
{
temp=check_touch();//检测触角的触碰情况
delay(2);
//延时 2 毫秒
//要使发挥 check_touch 的功能，必须延时
//延时时间决定甲虫触觉的灵敏度，延时越小，触觉越灵敏
switch(temp)
{
case 0x01://碰到右触角
led_process（1，1）；//右发光眼亮
motor_process(0,2);//后腿
motor_process(1,2);
delay(500);
motor_process(0,1);//右转
motor_process(1,0);
```

```
delay(100);
break;
case 0x04://碰到左触角
led_process（0，1）；//左发光眼亮
motor_process(0,2);//后退
motor_process(1,2);
delay(500);
motor_process(0,0);//左转
motor_process(1,1);
delay(100);
break;
default:
led_process（0，0）；//发光眼不亮
led_process（1，0）；
motor_process(0,1);//前进
motor_process(1,1);
break;
}
}
```

2.4 使你的 IQBug 更具个性

在开始设置你心爱的甲虫的个性之前，让我们先考虑一下它的能源问题。当甲虫的电压大于3.6 伏，甲虫可以正常工作；可是电压小于3.6 伏的时候，你可要小心了，甲虫很可能会出毛病了，这时候，你要给他充电了。我们可以调用函数 check_power 了解系统当前电压情况。

下面我们看看函数 check_power 的详细定义。

unsigned char check_power(void);

功能:检测系统当前电压值

返回值含义：系统当前电压值，例如：

36 表示 3.6V

42 表示 4.2V

例子：

电压不足：叫五声，通知你要给它充电了

躲避碰撞：

碰到左触角,后腿,右转，左发光眼亮；

碰到右触角,后腿,左转，右发光眼亮；

碰到左触角和右触角，后腿,左转，右发光眼亮

躲避强光：

左边光线强，走往右边；

右边光线强，走往左边

听你的指挥：如果听到声音，听到多少声，眨多少次眼

否则：前进

这个例子的源程序可参考文件 main.c.

好了,现在你可以开始设计你的甲虫的个性了。

2.5 深入了解你的 IQBug

假如你想对你心爱的甲虫有更多的了解，可参考我们的模块文件，里面有每一个模块的源代码以及详尽的说明。

不过，在看这些文件之前，你是否已经对AVR 单片机有一定的了解呢？假如还没有，那得先学习一下了，在学习的过程中要特别注意以下的一些内容：定时器、AD/DA 转换。

2.6 开发 IQBug 的潜能

到现在为止，你应该已经能够为甲虫设计出自己的个性了，那就可以继续开发甲虫的潜能了。

在一开始的时候，我们已经介绍了甲虫还具有串行通信和读写EEPROM 的能力。下面我们为你一一详细介绍。

unsigned char receive(unsigned char *data);

功能:把串口接受到的数据放到数据缓冲区 data

参数含义:

data: 数据缓冲区

返回值含义:

0：没有收到数据

1：有收到数据

void send(unsigned char data);

功能:通过串口把数据 data 发送出去

参数含义:

data: 要发送的数据

unsigned char x24c08_read(unsigned int address);

功能:读取 EEPROM 中地址为 address 的数据

参数含义:

address: EEPROM 中地址

返回值含义:

EEPROM 中地址为 address 的数据

void x24c08_write(unsigned int address,unsigned char info);

功能:把数据 info 写入 EEPROM，地址为 address

参数含义:

address: EEPROM 中地址

info：要写入 EEPROM 的数据

甲虫的这些功能是为了和我们在计算机上的控制软件上配合的，至于具体的情况，请参看第四章<<甲虫的计算机程序设计>>。

第三章 甲虫的结构设计

3.1 结构设计构想

甲虫的结构设计主要有两个出发点：

一、美观要求和功能要求：美观要求主要是要从人的审美观点出发，使得甲虫的外形结构尽最大可能的符合自然界的甲壳虫的的外观，同时加以拟人化进行适当的艺术提升，这就是我们设计甲虫外观的出发点。功能要求考虑主要是要从电器元件的布局来考虑，因为机械结构部件是甲壳虫受力的载体，同时也是电子元件布放的载体，所以在设计甲虫结构时既要从整体的受力分析来考虑，也要与电子设计工程师通力合作，设计好各种电子元件的布置空间。

二、安全法规和工艺要求：因为我们设计的产品主要是面向普通的个人使用，并且主要以儿童和未成年人居多，所以我们在设计这类产品必须考虑到国家以及国际的强制性安全因素，也就是我们的产品必须符合国家和国际的安全法规的要求。工艺要求主要是，我们设计的机械结构产品必须符合工艺要求，保证能够以低成本大批量供货。

3.2 部件材料综述

机器甲虫的机构部件主要由塑料零件和金属零件两类零件组成，除了少数连接固定用的螺钉等零件外，其余大部分零件由塑料零件组成。金属零件主要有弹簧和螺钉螺母等连接零件，这类零件主要材料是高性能的弹簧碳素钢以及不锈钢，而且都要经过最终表面处理，进行镀锌、镀镍等防腐蚀处理。

下面我们重点来探讨一下甲虫零件的主要材料-塑料。

塑料的成分主要由树脂、填充剂、增塑剂、着色剂、稳定剂、润滑剂等组成。塑料来源广泛，早已获得工业化大批量生产，广泛应用于航空、航天、机械、汽车、电子、电器、轻工、玩具等各个行业。塑料之所以使用广泛，因为它主要有以下优点：

（1） 重量轻。塑料是一种轻质材料，一般塑料的比重约在 0.83~2.2 之间，比钢铁等金属元件要轻的多。

（2） 比强度高。比强度是强度与密度之比，许多塑料的比强度相当高，其中玻璃纤维的比强度水平达到甚至超过了钢材的水平。

（3） 优良的耐磨、自润滑和吸震性能。

（4） 黏结能力强。一般塑料都有一定的黏结能力。

（5） 优越的化学稳定性。一般塑料对酸、碱、盐等化学药物均具一定的抗腐蚀能力。

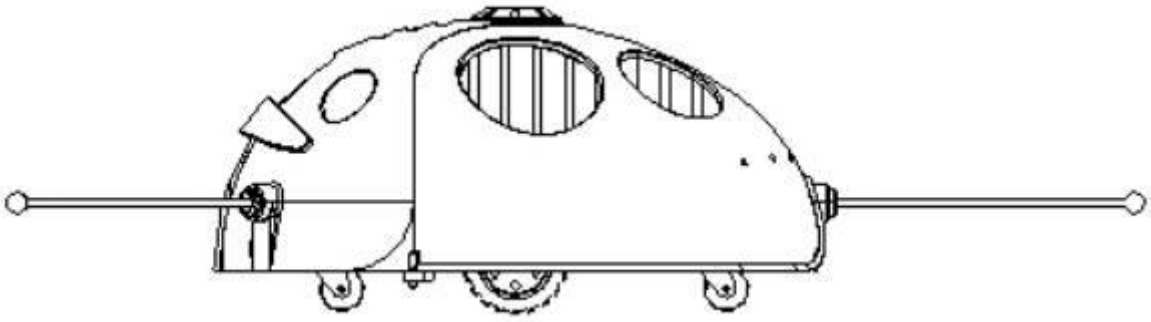
（6） 优良的电绝缘性能。一般塑料都是绝缘的，某些塑料甚至无论在高频还是低频、高压还是低压情况下，绝缘性能都十分优良。

当然，话又说回来，塑料并非完美无缺，塑料不能耐高温，在高温加热和光线照射下，容易分解、老化，另外，塑料的使用寿命相对金属来说要短。

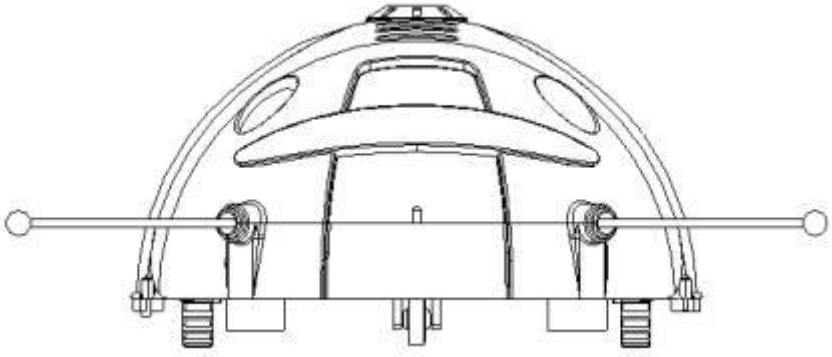
塑料有很多种类，目前，已知投产的塑料品种有300 多种。在我们这里，机器甲虫的机械部件所采用的塑料品种是 ABS，ABS 塑料是目前全球产量最大，应用最广的一种工程塑料，它具有综合性的优良性能，坚固、坚韧、坚硬，而且无毒无味等。

3.3 外形设计简述

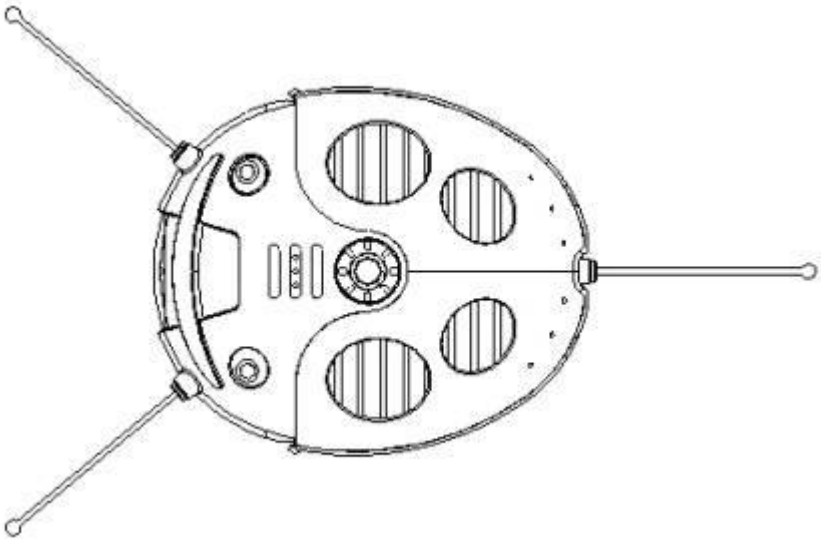
甲虫的基本外形的侧面投影轮廓线为一段直线和一段平滑过度的曲线组成，正面投影为一段直线和一段半圆弧组成，由此我们得知甲虫的基本外形为一曲线回转体的一半。这样的曲面造型美观，容易实现加工。外观图如下：



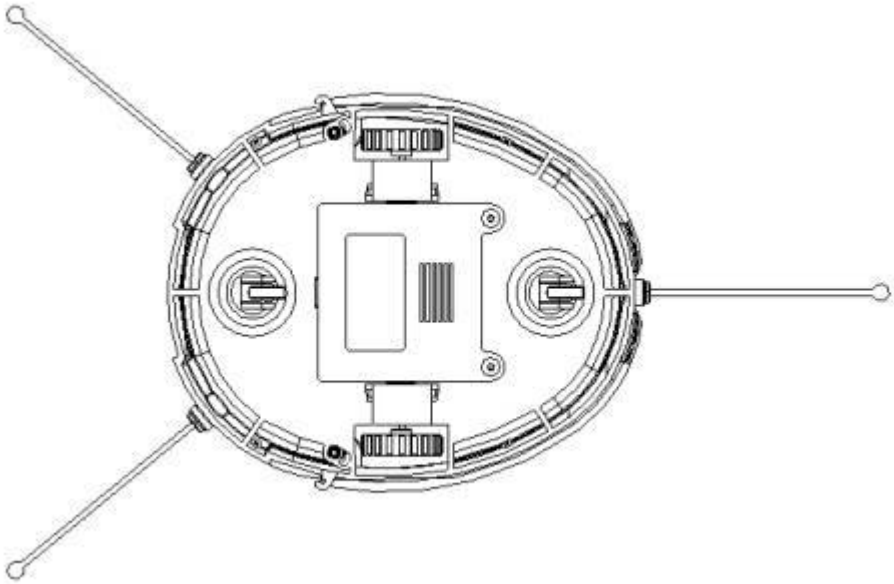
（一）侧面投影图



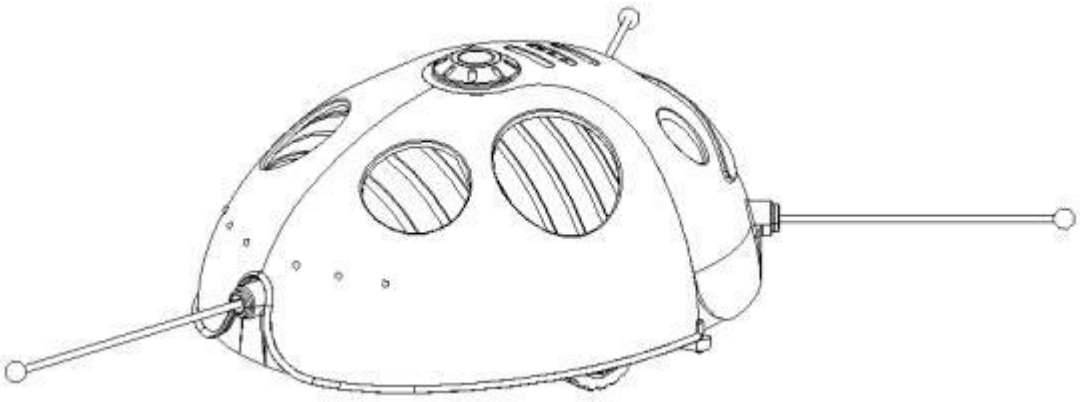
（二）正面投影图



（三）俯视投影图

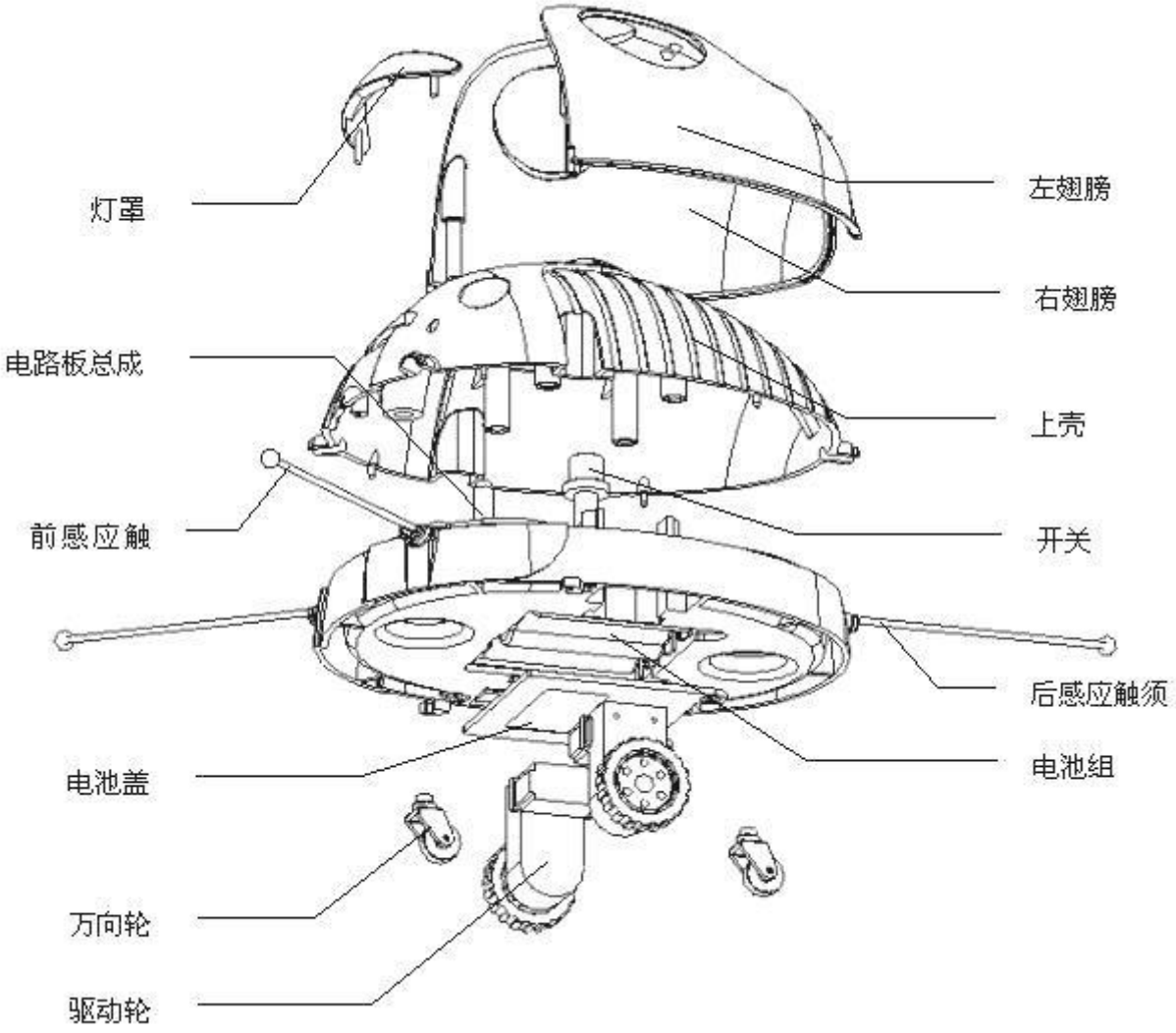


（四）仰视投影图

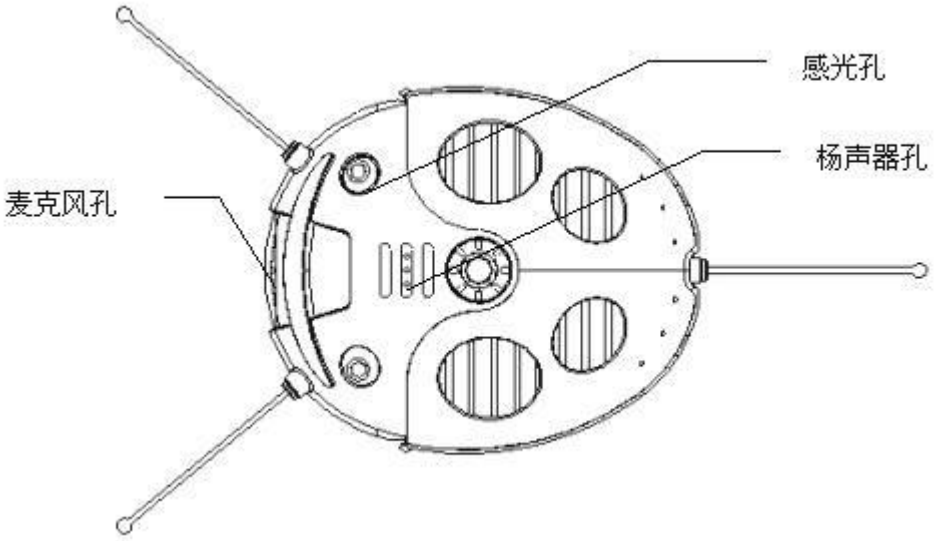


（五）等角投影图

1. 装配图零件认识



（爆炸图）



(俯视图)

第四章 甲虫的计算机程序设计

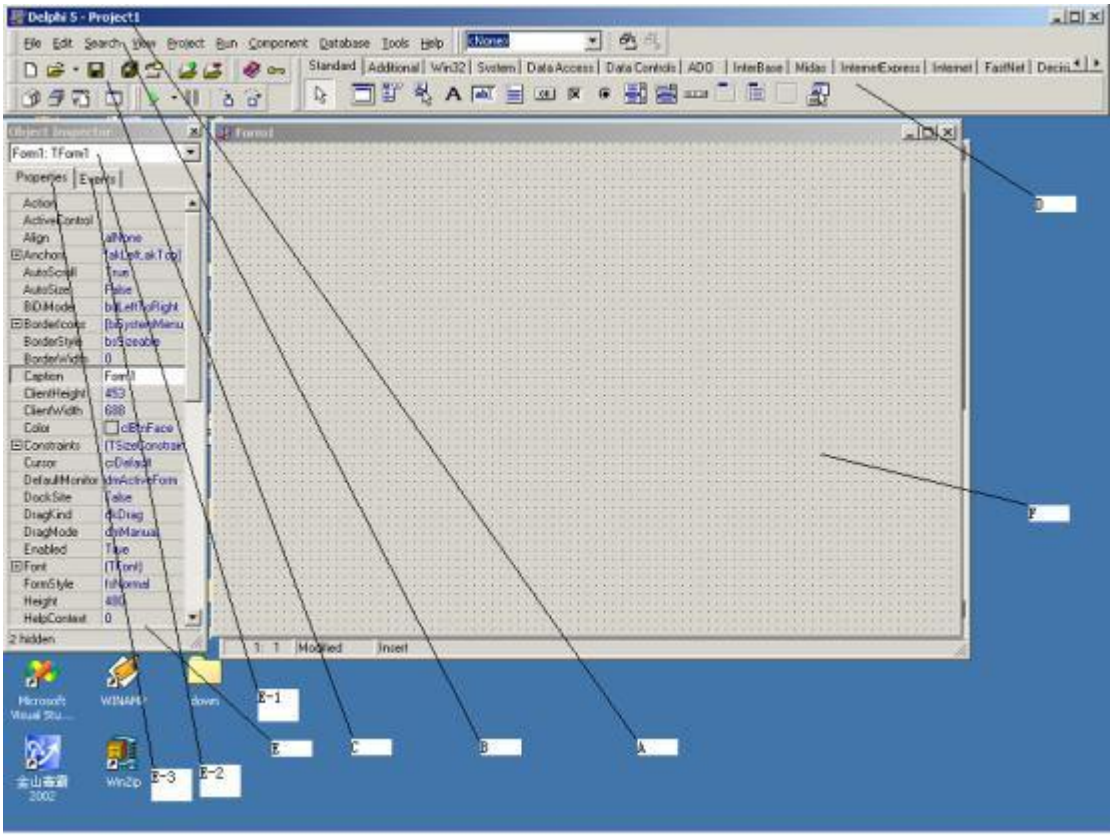
怎么用 Delphi 编写机器甲虫的控制软件

机器人这个名字,对大家来说应该不陌生了.随着机器人的时代来临,这种潮流的呼声,人们对机器人也越来越重视了.本文是我在开发机器甲虫控制软件的一些流程和代码,想以此来让读者掌握 Delphi 开发工具及机器人控制软件的编写.

4.1 Delphi 开发环境

首先,介绍一下 Delphi 吧,Delphi 是 Inprise 公司(原 Borland 公司)的面向对象的可视化软件开发工具.目前,还有很多其他软件开发工具,比如 Visual C++和 Visual Basic 等.一般情况下,学习和使用 Visual C++是”先苦后甜”:Visual C++的功能强大,但是上手比较困难;学习和使用 Visual Basic 的过程是”先甜后苦”:Visual Basic 容易上手,但是功能有限.Delphi 集中了这两者的优点:学习过程很容易上手,而且功能也非常强大.而且流传这么一句话吧,真正的程序员用 C,聪明的程序员用 Delphi!

说到这里,大家一定想见见 Delphi 是什么样子呢?好吧,就介绍一下 Delphi 5(现在已经有 Delphi 7 开发环境基本上相似),当你打开 Delphi 时一定出现以下这个界面(也就是主界面):



(图 4-1-1)

图中 A 是标题栏:显示是当前的工程名,它位于最上部了.

图中 B 是主菜单:是开发环境中发绝大部分命令的地方.这个地方可要注意学习了.建议大家要试试所有菜单中的命令了.

图中 C 是工具栏:是将菜单中的一些常用命令设定在这里,便于开发者的使用.

图中 D 是组件栏:组件的集合

图中 E 是对象编辑器:是实现对象(组件)的属性设置,创建事件处理过并进行管理.它包括:

E-1:对象列表:当前窗体上所有的组件集合.有时一些组件在窗体不容易选定,可能用选定.

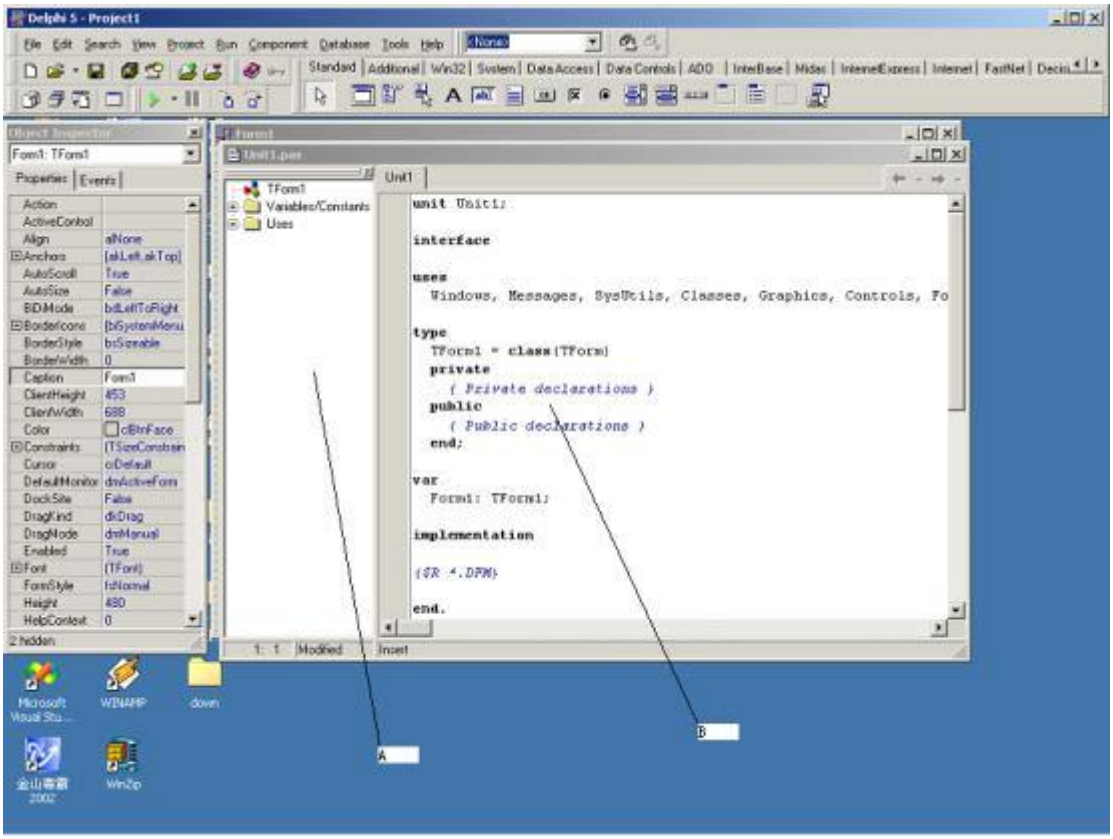
E-2:事件页

E-3:属性页

图中 F 是窗体设计器:是用来为设置软件的窗体提供快速的利器.

说到这里大家一定发现了窗体设计器后面有一个窗体吧.好了,让我们看看吧.怎么看呢?

好吧,我们一起来看吧,用鼠标点一下窗体设计器,再按一下 F12 出现了什么呢?是不是下图呢?



(图 4-1-2)

好吧,图 4-1-2 中的 A 是:代码浏览器,通过它可以快速地定位.

图 4-1-2 中的 B 是:代码编辑辑器,为代码输入和编辑提供了一个方便的环境.

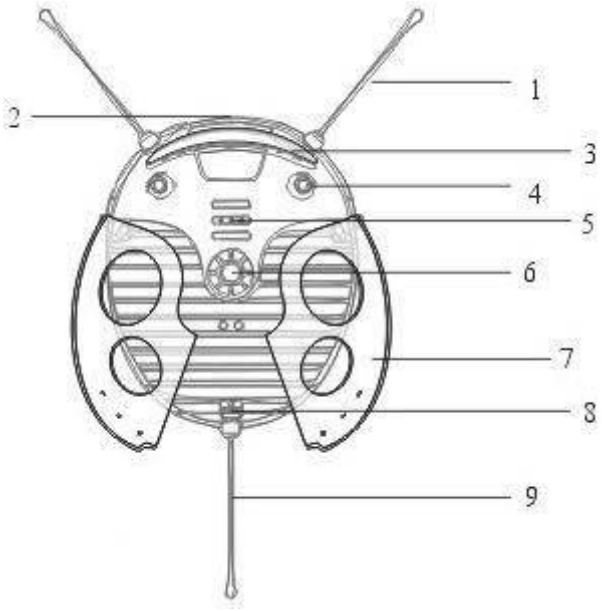
4.2 软件设计

4.2.1 机器甲虫功能分析(需求分析与功能分析及协议)

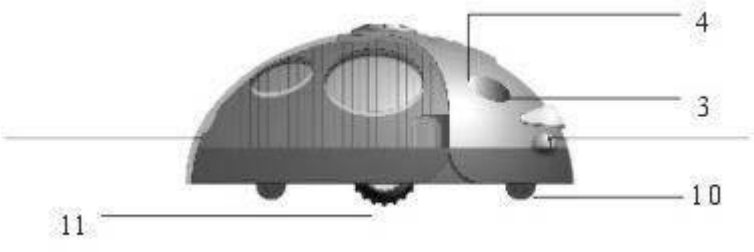
好了,前面说了那么多.大家对 Delphi 一定有印象了吧.大家一定知道我现在要说的是什么呢?机器甲虫是怎么工作的呢?

自然的甲虫大家应该都看过吧,它可是动物中很低能的.我们知道甲虫它有美丽的翅膀,小眼睛,秀长的脚,长长的触须.它们成自然界一道亮丽的风景,你们一定会说,这是自然界的啊,那让你们先看看我们的机器甲虫吧!那会是什么样子呢?

(图 4-2-1)



(图 4-2-2)



(图 4-2-3)

看了吧,图 4-2-1 是整体外观图.图 4-2-2 和图 4-2-3 是结构剖析图.让我们一起来看看吧:

1 为前触须,是不是有点自然的触须呢?对,就是根据它来设计的.它是用感知前面是否有物体的

2. 耳朵,他的耳朵有怪异了吧.可是你放心,它是能绝对听到声音的.

3. 会发光的眼睛:听到这个名字有点奇怪吧,是的,这可能与自然界有点不同,它就是只会发光不能看见东西..所以就叫会发光的眼睛了.但看得见的眼睛在后面呢.

4. 会感光的眼睛:这就是我要跟你说看得见的眼睛了,它能感觉光的强弱.也称为光敏.

5. 嘴巴.甲虫的嘴巴只会发几种声音了.可我们的机器甲虫就利害了.它还会唱很多歌呢.

6. 电源开关

7. 翅膀

8. 电脑通讯插口,这是可是机器甲虫学习的通道了.

9. 尾巴

10. 万向转动轮

11. 小腿驱动轮,这就甲虫的脚了.

现在大家已经知道了.机器甲虫的结构了.但是他是怎么工作和运动的呢?

大家可能已经看出来.前面结构介绍中.怎么没有大脑呢?是的,机器甲虫就是用大脑来指挥各种器官行动以及器官的反应.大脑在机器甲虫里面,在外面是看不见的.这里的大脑也就是我们所说的CPU.它能快速地作出各种指令和处理器官传给的指令.

大脑是怎么工作的呢?很简单,就是将个器官组合起来工作就行了.这就是我们的单片机编程了.要知道它是怎样实现的可要下点功夫.

一个大脑设计好了以后,他开始只有一个结构在哪里.各种智能还要你去教它呢?你一定会讲我怎么跟它沟通呢?好吧,我们先别说这个,就是我们人之间的沟通吧.我们会用彼此之间都知道的语言来交流吧.你同一个外国人沟通,那你一定要知道说外国话.才能交流对不?那跟机器甲虫交流也要知道它的语言啊.这就是我们说的协议了.

刚才我们说了这么多,知道了机器甲虫的工作原理了吧.

有了机器甲虫,我们想做些什么呢?这就是我们的需求了.现在我就说说我们的需求吧.

1. 左右眼能发光

2. 能从嘴发声

3. 脚的前后运动.

4. 光敏反应

5. 听声响的反应

6. 左右触角的反应
7. 尾巴的反应.

有了这些需求,我们就以这些需求来规范化们的功能吧.

1. PC 机控制左右眼的发光
2. PC 机控制嘴的发声
3. PC 机控制左右脚的运动
4. 机器甲虫的光敏反应在 PC 机上显示出来
5. 听声响的反应在 PC 机上显示出来
6. 左右触角的反应在 PC 机上显示出来
7. 尾巴的反应在 PC 机上显示出来

好了,现在我们来看看这些功能吧.我们可分为两种类型.一种是从 PC 上发指令控制机器甲虫.一种是机器甲虫对环境的反应在 PC 机上让用户知道.

4.2.2 界面设计

在软件开发中,界面的设计是必不可少的,而且是重要的一步,它的设计与软件功能,软件的用户等众多的因素来设计的,但这些原则是必须遵循的,界面友好,用户操作方便,简便易用.下面就是我们要设计的界面了,如图 4-2-4

(图 4-2-4)

指令一栏是PC机控制机器甲虫,左边的三个复选框代表左右眼和嘴巴.如果选定则表示相应的左右眼发光,嘴发声.左右脚选定机器甲虫则会向前走或向后走.

反应一栏是机器甲虫的反应在PC机上显示出来.光敏左右表示机器甲虫的左右眼的光线强度.听声响表示听到一次声响还是二次声响或者没有,左右触角表示左右触角是否碰到东西,尾巴表示尾巴是否碰到东西.

4.3 软件实现与代码编写

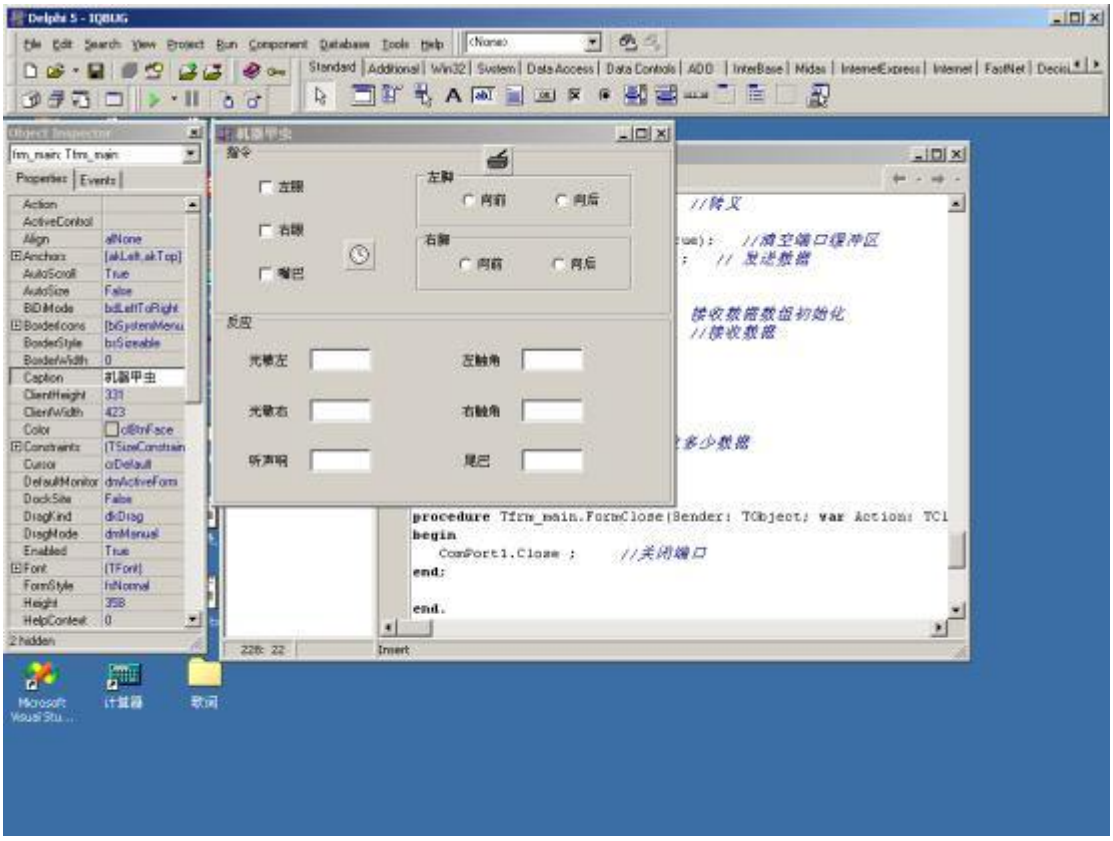
现在我们一起来一步一步的完成我们的程序编码吧.

4.3.1 新建一个工程(IQBUG)

打开DELPHI,单击文件菜单FILE->NEW APPLICATION,这时就新建了一个空的工程了,我们可以这时就把它保存起来.单击菜单FILE->SAVE ALL,会出现SAVE对话,选择保存的路径,然后取一个文件名为:U_main吧.点击保存按钮,则出现在另一个SAVE对话框,同样选择好路径.取一个文件名为IQBUG吧.点击保存按钮.刚才我们保存了两个文件.哪是什么呢?我们在新建一个工程时,DELPHI 会自动帮你产生一个单元,所以前面的就是自动产生的单元.后一个才是我们工程文件.

4.3.2 设计界面

现在我们可以先用组件可以很方便的设计我们的界面了.我们先把我们的窗体命令吧!在对象编辑器中将窗体FORM1的NAME属性改为FRM_MAIN.Caption属性设为”机器甲虫”.再在这个窗体中增加以下组件.如图4-3-1.



(图 4-3-1)

GroupBox1: TGroupBox;
它的一些属性如下:
Left = 0
Top = 0
Width = 423
Height = 157
Align = alTop
Caption = ' 指令 '

GroupBox2: TgroupBox
它的一些属性如下:
Left = 0
Top = 157
Width = 423
Height = 174
Align = alClient
Caption = ' 反应 '

GroupBox3: TgroupBox
它的一些属性如下:
Left = 186
Top = 23
Width = 200
Height = 54
Caption = ' 左脚 '

GroupBox4: TgroupBox
它的一些属性如下:
Left = 186
Top = 81
Width = 200
Height = 54
Caption = '右脚'

CkLeftEye: TcheckBox
它的一些属性如下
Left = 41
Top = 31
Width = 57
Height = 17
Caption = '左眼'

CkRightEye: TcheckBox
它的一些属性如下:
Left = 41
Top = 71
Width = 57
Height = 17
Caption = '右眼'

CkMouth: TcheckBox
它的一些属性如下:
Left = 41
Top = 111
Width = 57
Height = 17
Caption = '嘴巴'

RDLeftLegFw: TradioButton
它的一些属性如下:
Left = 42
Top = 20
Width = 49
Height = 17
Caption = '向前'

RDLeftLegBa:TradioButton
它的一些属性如下:
Left = 128
Top = 20
Width = 49
Height = 17
Caption = '向后'

RDRightLegFw:TradioButton
它的一些属性如下:
Left = 39
Top = 21
Width = 49
Height = 17
Caption = '向前'

RDRightLegBa:TRadioButton
它的一些属性如下:
Left = 125
Top = 21
Width = 49
Height = 17
Caption = '向后'

Label3: Tlabel
Left = 32

Label4: TLabel
Left = 32

Label5: TLabel
Left = 32

Label6: TLabel
Left = 229

```
    Top = 35                Top = 82                Top = 128               Top = 35
    Width = 36              Width = 36              Width = 36              Width = 36
    Height = 13             Height = 13             Height = 13             Height = 13
    Caption = '光敏左'       Caption = '光敏右'       Caption = '听声响'       Caption = '左触角'

Label7: TLabel          Label8: TLabel
    Left = 229              Left = 231
    Top = 82                Top = 128
    Width = 36              Width = 24
    Height = 13             Height = 13
    Caption = '右触角'       Caption = '尾巴'

EdtLeftLG: TEdit        EdtRightLG: TEdit       EdtSound: TEdit         EdtLeftFE: TEdit
      Left = 88               Left = 88               Left = 88               Left = 284
      Top = 31                Top = 78                Top = 124               Top = 31
      Width = 57              Width = 57              Width = 57              Width = 57
      Height = 21             Height = 21             Height = 21             Height = 21

EdtRightFE: TEdit       EdtTail: TEdit
      Left = 284              Left = 284
      Top = 78                Top = 124
      Width = 57              Width = 57
      Height = 21             Height = 21

Timer1: TTimer          ComPort1: TComPort
    Left = 120              BaudRate = br9600
    Top = 88                Port = 'COM1'
```

4.3.3 代码编辑

是不是很轻松地把界面设计好了吧!对了现在接下来的是一步很重要的工作了.哪就是代码编辑.
首先设定两个成员数组,用来窜口发数据,各收数据的.在代码编辑器中的

```
private
    { Private declarations }后增加
    sendbuf:array[0..30] of BYTE; //发送窜口数据
    recbuf:array[0..30] of BYTE; //接收窜口数据
```

第二步,写 FORM 的 SHOW 事件

```
procedure Tfrm_main.FormShow(Sender: TObject);
begin
    ComPort1.Port:='COM1'; //设定通信端口为 COM1
    ComPort1.Open ; //打开端口
    CkLeftEye.Checked:=False; //左眼初始为灭
    CkRightEye.Checked:=False; //右眼初始为灭
    CkMouth.Checked:=False; //嘴初始为不发声
```

```
    EdtLeftLG.Clear; //光敏左初始化
    EdtRightLG.Clear; //光敏右初始化
    EdtSound.Clear; //听声响初始化
    EdtLeftFE.Clear; //左触角初始化
    EdtRightFE.Clear; //右触角初始化
    EdtTail.Clear; //尾巴初始化
    sendbuf[0]:=$7e;
    sendbuf[1]:=$c2;//电脑对机器甲虫对话的起始符
end;
```

第三步.时钟 TIMER1 事件.

```
procedure Tfrm_main.Timer1Timer(Sender: TObject);
var
   isum,i,ii:integer;

   procedure explain;  //解释接收到的数据
   var
     posb:Array[0..9] of BYTE ;//转义后收到的数据
   begin
      posb[0]:=recbuf[0] ;
      posb[1]:=recbuf[1] ;
      i:=2 ;
      ii:=2;
      while (i<9) do
      begin
        if recbuf[ii]=$7d then
        begin
          if recbuf[ii+1]=$5e then
          begin
             posb[i]:=$7e ;
             ii:=ii+1 ;
          end
          else if recbuf[ii+1]=$5d then
          begin
             posb[i]:=$7d ;
             ii:=ii+1 ;
          end;
        end
        else
```

```
          posb[i]:=recbuf[ii] ;
        i:=i+1 ;
        ii:=ii+1;
      end;      //转义
      if posb[8]=(posb[0]+posb[1]+posb[2]+posb[3]+posb[4]+posb[5]+posb[6]+posb[7]) mod 256 then
      begin
         if posb[2]>0 then
         begin
            EdtRightFE.Text:='动' ;
         end
         else
           EdtRightFE.Clear;
         if posb[3]>0 then
         begin
            EdtTail.Text:='动';
         end
         else
            EdtTail.Clear;
         if posb[4]>0 then
         begin
            EdtLeftFE.Text:='动';
         end
         else
            EdtLeftFE.Clear;
         EdtSound.Text:=intToStr(posb[5]) ;
         EdtLeftLG.Text:=IntTOStr(posb[6] div 8) ;
         EdtRightLG.Text:=IntToStr(posb[7]div 8) ;
      end;
   end;
begin
   if CkLeftEye.Checked then
   begin
      sendbuf[2]:=$01;//左眼 亮为$01 灭为$FF
      sendbuf[3]:=$FF; //亮多久
   end
   else begin
      sendbuf[2]:=$FF;
      sendbuf[3]:=$FF;
   end;
   if CkRightEye.Checked then
```

```
begin
   sendBuf[4]:=$01;//右眼 亮为$01 灭为$FF
   sendbuf[5]:=$FF;//亮多久
end
else begin
   sendbuf[4]:=$FF;
   sendbuf[5]:=$FF;
end;
if CkMouth.Checked then
begin
   sendbuf[6]:=161;       //嘴是否发声
   sendbuf[7]:=$FF;
end else
begin
  sendbuf[6]:=$ff;
  sendbuf[7]:=$FF;
end;
if RDLeftLegFw.Checked then
begin
   sendbuf[8]:=$01;   //左脚向前走
   sendbuf[9]:=$FF;
end
else if RDLeftLegBa.Checked then
begin
   sendbuf[8]:=$02;    //左脚向后走
   sendbuf[9]:=$FF;
end
else begin
   sendbuf[8]:=$FF;     //左脚不动
   sendbuf[9]:=$FF;
end;
if RDRightLegFw.Checked then
begin
   sendbuf[10]:=$01;  //右脚向前走
   sendbuf[11]:=$FF;
end
else if RDRightLegBa.Checked then
begin
   sendbuf[10]:=$02;    //右脚向后走
   sendbuf[11]:=$FF;
```

```
    end
    else begin
       sendbuf[10]:=$FF;     //右脚不动
       sendbuf[11]:=$FF;
    end;

    isum:=0 ;
    for i:=0 to 11 do
      isum:=isum+sendbuf[i];
    sendbuf[12]:=isum mod $100;     //简单校验和
    isum:=13;

    if sendbuf[12]=$7e then
    begin
       sendbuf[12]:=$7d;
       sendbuf[13]:=$5e;
       isum:=14;
    end
    else if sendbuf[12]=$7d then
    begin
       sendbuf[12]:=$7d;
       sendbuf[13]:=$5d;
       isum:=14;
    end;                          //转义

    ComPort1.ClearBuffer(True,True);   //清空端口缓冲区
    ComPort1.Write(sendbuf,isum);   // 发送数据

    for i:=0 to 30 do
       recbuf[i]:=0;          //  接收数据数组初始化
    ComPort1.Read(recbuf,30);    //接收数据
    for i:=30 downto 0 do
    if recbuf[i]<>0 then
    begin
      isum:=i+1;
      break;
    end;              //判断接收多少数据
    explain;         //解释数据
end;
```

第四步,关闭窗体时关闭端口.

```
procedure Tfrm_main.FormClose(Sender: TObject; var Action: TCloseAction);
begin
     ComPort1.Close ; //关闭端口
end;
```

4.3.4 编译运行

代码编好后,我们就可以编译成可执行文件了.按 CTRL+F9 键(或点击菜单 Project->Compile IQBUG)这样就编译好了.这时在工程文件的目录会生成了一个 IQBUG.exe 文件.这就是可执行文件了.也就是你的机器甲虫控制软件了.试试看怎么样呢?

附录

附录一 Object Pascal 语言

注解

Object Pascal 支持三种类型的注解：花括号注解、圆括号/星号注解和双斜杠注解。下面是每种类型的例子：

```
{花括号注解}
( *圆括号/星号注解* )
//双斜杠注解
```

前两种注解在本质上是相同的，编译器把在注解限定符头和注解限定符尾中间的内容当作注解。对于双斜杠注解来说，双斜杠后面到行尾的内容被认为是注解。

注意：相同类型的注解不能嵌套。虽然不同类型的注解进行嵌套在语法上是合法的，但我们不建议这样做。这里是一些例子：

```
{ ( *这是合法的* ) }
( * {这是合法的} * )
( * ( *这是非法的* ) * )
{ {这是非法的} }
```

数据类型

对象 Pascal 的数据类型比较丰富.现在我就按类介绍一些常用的数据类型吧.

简单数据类型

1. 整数类型

   8 位有符号整数 ShortInt 数值范围:-128~127

   8 位无符号整数 Byte 数值范围:0~255

   16 位有符号整数 SmallInt 数值范围:-32768~32767

   16 位无符号整数 Word 数值范围:0~65535

   32 位有符号整数 Integer,LongInt 数值范围:-2147483648~2147483647

   32 位无符号整数 Cardinal,LongWord 数值范围:0~4294967295

2. 字符类型

   1 字节字符 AnsiChar,Char

   2 字节字符 WideChar

3. 布尔类型

   单字节 Boolean,ByteBool

   双字节 WordBool

   四字节 LongBool

   该类型只有两个取值,为真时取 True,为假时取 False

4. 枚举类型

   是由一组有序的标识符组成.形式如下:

   type typename=(value1,…,value2)

   例如:颜色吧

   type Tcolors=(Red,Blue,Green,Yellow,Orange,Purple,White,Black);//类型定义

   var my_color:Tcolors;//变量声明

5. 子界类型

   它为某个有序类型的子集.例如上面我定义了一个颜色的枚举类型,下面我们就定义一个子界类型吧:

   type TmyColors=Blue..Orange;

   那么这个子界类型包括了:Blue,Green,Yellow,Orange.

6. 实数数据类型

   4 字节浮点数 Single 数值范围 1.5*10^-45..3.4*10^38

   6 字节浮点数 Real48 数值范围 2.9*10^-9..1.7^10^38

   8 字节浮点数 Double 数值范围 5.0*10^-324..1.7*10^308

   10 字节浮点数 Extended 数值范围 3.6*10^-4951..1.1*10^4932

字符串类型

1. ShortString 最大长度为 255 存储空间为 2 到 256 字节
2. AnsiString 最大长度为 2^31 存储空间为 4 字节到 2GB
3. WideString 最大长度为 2^30 存储空间为 4 字节到 2GB

结构类型

1. 集合类型

集合类型(set)由有序类型的一些数值组成.定义方法:set of baseType

```
type
TSomeInts=0..255; //基本类型为有序类型
TintSet=set of TSomeInts //定义集合类型
```

2. 数组类型

数组类型(array)的数据为某一类相同类型的元素按一定的顺序组成的序列,包括静态数组和和动态数组两种.

静态数组：array[indexType1,…,indexTypen]of baseType

例：

```
var MyArray:array[0..30] of Integer; //一维数组 (31 个整数组成的数组)
var MyMulArray:array[0..10,0..20] of Byte; //二维数组
```

动态数组：array of baseType

例：

```
var MyIntArray:array of Integer; //定义动态数组
```

可以通过过程 SetLength 来为动态数组指定空间大小.

```
SetLength(MyIntArray,10); //指定动态数组空间大小.
```

3. 记录类型

记录类型(record)可以由不同类型的元素组成,这些元素称为域.定义格式:

```
Type recordTypeName=record
    FieldList1:Type1;
    …
    FieldListn:Typen;
End;
```

例:

```
type stugrade=record
    stu_no:integer; //学号
    stu_na:string; //姓名
    ch_num:real; //语文分数
    ma_num:real; //数学分数
end;
```

运算符

1. 算术运算符

运算符	作 用	操作数类型	结果类型
+	两个数相加	Integer,real	Integer,real
-	两个数相减	Integer,real	Integer,real

*	两个数相乘	Integer,real	Integer,real
/	两个浮点数相除	Integer,real	Real
Div	两个整数相除	Integer	Integer
mod	取模	Integer	Integer

2. 逻辑运算符

not 逻辑取反运算

and 逻辑和运算

or 逻辑或运算

xor 逻辑异或运算

3. 位运算符

位运算符对整型的数据进行按位操作,看得结果也是整数

not 按位取反

and 按位取和

or 按位或

xor 按位异或

shl 按位左移

shr 按位右移

4. 关系运算符

= 判断是否相等

<> 判断是否不相等

< 判断是否小于

> 判断是否大于

<= 判断是否小于等于

>= 判断是否大于等于

语句语法

1. 声明语句

声明语句包括对标号,常量,变量,数据类型,过程,函数等声明.以下是几种常见的声明格式如下:

标号声明:

```
Label Label1,…,Labeln;常量声明:const PI=3.1415927;
```

变量声明:

```
var
    I,ii:integer;
```

```
    Mystr:string;
```

2. 赋值语句

赋值语句形式: :=;

例:

```
posi:=12; //将变量 posi 赋值为 12
```

3. goto 语句

通过该语句可以在程序中非常方便地从一个地方直接跳转到另一个地方.但建议尽量不要使用 goto 语句.在你使用 goto 语句之前,要先声明一个标号.

形式:goto label;

4. with 语句

形式: with obj do statement

例:

```
with Edit1 do
begin
    Clear;
    Text:=’123’;
    SetFocus;
end;
```

相当于:

```
Edit1.Clear;
Edit1.Text:=’123’; 028-87428032
Edit1.SetFocus;
```

5. if 语句

if 语句主要实现判断的功能.

形式为:if expression then statement;

中文意思为: 如果 真 然后 执行语句

例:

```
if ipos>10 then
    ipos=0
else
    ipos:=ipos+1 ;
```

上面语句的意思是:如果变量 ipos 大于 10 那么 ipos 等于 0;否则,ipos 加 1;

6. case 语句

case 语句对一个结果的多个分支进行判断,形式为:

```
case selectorExpression of
 caseList1:statement1;
 …
 caseListn:statementn;
```

```
else
     statement;
end;
```

例:

```
case ire of
     1:strstate:=’ Blue’ ;
     2:strstate:=’ Green’ ;
     3:strstate:=’ Red’ ;
else
     strstae:=’ Gray’ ;
end;
```

上面语句相当于:

```
if ire=1 then
   strstate:=’ Blue’
else if ire=2 then
        strstate:=’ Green’
     else if ire=3 then
             strstate:=’ Red’
          else
             strstate:=’ Gray’ ;
```

7. repeat 语句

repeat 语句是循环语句.形式为:

```
repeat
   循环体
until
   expression
```

在表达式 expression 为 True 之前,循环执行循环体.

例:

```
repeat
   inc(i);
until i>100;
```

8. while 语句

这也是循环语句.形式为:

```
while expression do statement
```

在表达式 expression 为 False 之前,循环执行 statement;

例:

```
while i<=100 do
begin
```

```
            inc(i)
        end;
```

9. for 语句

for 语句也是循环语句.形式为:

```
for counter:=initialvalue to finalvalue do statement
```

或

```
for counter:=initlialvalue downto finealvalue do statement
```

```
for I:=1 to 100 do
begin
    isum:=isum+i;
end;
```

附录二 Delphi 参考手册

名称	类型	说明
Abort	函数	引起放弃的意外处理
Abs	函数	绝对值函数
AddExitProc	函数	将一过程添加到运行时库的结束过程表中
Addr	函数	返回指定对象的地址
AdjustLineBreaks	函数	将给定字符串的行分隔符调整为 CR/LF 序列
Align	属性	使控件位于窗口某部分
Alignment	属性	控件标签的文字位置
AllocMem	函数	在堆栈上分配给定大小的块
AllowGrayed	属性	允许一个灰度选择
AnsiCompareStr	函数	比较字符串（区分大小写）
AnsiCompareText	函数	比较字符串（不区分大小写）
AnsiLowerCase	函数	将字符转换为小写
AnsiUpperCase	函数	将字符转换为大写
Append	函数	以附加的方式打开已有的文件
ArcTan	函数	余切函数
AssignFile	函数	给文件变量赋一外部文件名
Assigned	函数	测试函数或过程变量是否为空
AutoSize	属性	自动控制标签的大小
BackgroundColor	属性	背景色
BeginThread	函数	以适当的方式建立用于内存管理的线程
BevelInner	属性	控件方框的内框方式
BevelOuter	属性	控件方框的外框方式
BevelWidth	属性	控件方框的外框宽度
BlockRead	函数	读一个或多个记录到变量中
BlockWrite	函数	从变量中写一个或多个记录
BorderStyle	属性	边界类型
BorderWidth	属性	边界宽度
Break	命令	终止 for、while、repeat 循环语句
Brush	属性	画刷

Caption	属性	标签文字的内容
ChangeFileExt	函数	改变文件的后缀
ChDir	函数	改变当前目录
Checked	属性	确定复选框选中状态
Chr	函数	返回指定序数的字符
CloseFile	命令	关闭打开的文件
Color	属性	标签的颜色
Columns	属性	显示的列数
CompareStr	函数	比较字符串（区分大小写）
Concat	函数	合并字符串
Continue	命令	继续 for、while、repeat 的下一个循环
Copy	函数	返回一字符串的子串
Cos	函数	余弦函数
Ctl3D	属性	是否具有 3D 效果
Cursor	属性	鼠标指针移入后的形状
Date	函数	返回当前的日期
DateTimeToFileDate	函数	将 DELPHI 的日期格式转换为 DOS 的日期格式
DateTimeToStr	函数	将日期时间格式转换为字符串
DateTimeToString	函数	将日期时间格式转换为字符串
DateToStr	函数	将日期格式转换为字符串
DayOfWeek	函数	返回星期的数值
Dec	函数	递减变量值
DecodeDate	函数	将日期格式分解为年月日
DecodeTime	函数	将时间格式分解为时、分、秒、毫秒
Delete	函数	从字符串中删除子串
DeleteFile	命令	删除文件
DiskFree	函数	返回剩余磁盘空间的大小
DiskSize	函数	返回指定磁盘的容量
Dispose	函数	释放动态变量所占的空间
DisposeStr	函数	释放字符串在堆栈中的内存空间
DitherBackground	属性	使背景色的色彩加重或减少 50%
DragCursor	属性	当鼠标按下时光标的形状
DragMode	属性	按动的作用方式
DropDownCount	属性	容许的显示数据项的数目
EditMask	属性	编辑模式
Enabled	属性	是否使标签呈现打开状态
EncodeDate	函数	将年月日合成为日期格式
EncodeTime	函数	将时、分、秒、毫秒合成为时间格式
EndMargin	属性	末尾边缘
Eof	函数	对有类型或无类型文件测试是否到文件尾
Eoln	函数	返回文本文件的行结束状态
Erase	命令	删除外部文件
ExceptAddr	函数	返回引起当前意外的地址
Exclude	函数	从集合中删除一些元素
ExceptObject	函数	返回当前意外的索引
Exit	命令	立即从当前的语句块中退出
Exp	函数	指数函数
ExpandFileName	函数	返回包含绝对路径的字符串

ExtendedSelect	属性	是否允许存在选择模式，True 时，MultiSelect 才有意义
ExtractFileDir	函数	返回驱动器和路径
ExtractFileExt	函数	返回文件的后缀
ExtractFileName	函数	返回文件名
ExtractFilePath	函数	返回指定文件的路径
FileAge	函数	返回文件已存在的时间
FileClose	命令	关闭指定的文件
FileCreate	命令	用指定的文件名建立新文件
FileDateToDateTime	函数	将 DOS 的日期格式转换为 DELPHI 的日期格式
FileExists	函数	检查文件是否存在
FileGatAttr	函数	返回文件的属性
FileGetDate	函数	返回文件的 DOS 日期时间标记
FileOpen	命令	用指定的存取模式打开指定的文件
FilePos	函数	返回文件的当前指针位置
FileRead	命令	从指定的文件读取
FileSearch	命令	在目录中搜索指定的文件
FileSeek	函数	改变文件的指针
FileSetAttr	函数	设置文件属性
FileSetDate	函数	设置文件的 DOS 日期时间标记
FileSize	函数	返回当前文件的大小
FileWrite	函数	对指定的文件做写操作
FillChar	函数	用指定的值填充连续字节的数
FindClose	命令	终止 FindFirst/FindNext 序列
FindFirst	命令	对指定的文件名及属性搜索目录
FindNext	命令	返回与文件名及属性匹配的下一入口
FloatToDecimal	函数	将浮点数转换为十进制数
FloatToStrF	函数	将浮点数转换为字符串
FloatToStr	函数	将浮点数转换为字符串
FloatToText	函数	将给定的浮点数转换为十进制数
FloatToTextFmt	函数	将给定的浮点数转换为十进制数
Flush	函数	将缓冲区的内容刷新到输出的文本文件中
FmtLoadStr	函数	从程序的资源字符串表中装载字符串
FmtStr	函数	格式化一系列的参数，其结果以参数 Result 返回
Font	属性	设置字体
Format	函数	格式化一系列的参数并返回 Pascal 字符串
FormatBuf	函数	格式化一系列的参数
FormatDateTime	函数	用指定的格式来格式化日期和时间
FormatFloat	函数	指定浮点数格式
Frac	函数	返回参数的小数部分
FreeMem	函数	按给定大小释放动态变量所占的空间
GetDir		返回指定驱动器的当前目录
GetHeapStatus		返回内存管理器的当前状态
GetMem		建立一指定大小的动态变量，并将指针指向该处
GetMemoryManager		返回内存管理器的入口点
Glyph	函数	按钮上的图象
Halt		停止程序的执行并返回到操作系统
Hi		返回参数的高地址位
High		返回参数的上限值

Hint	属性	提示信息
Int		返回参数的整数部分
Include		添加元素到集合中
Insert		在字符串中插入子串
IntToHex		将整型数转换为十六进制数
IntToStr		将整型数转换为字符串
IOResult		返回最新的 I/O 操作完成状态
IsValidIdent		测试字符串是否为有效的标识符
Items	属性	默认显示的节点
Kind	属性	摆放样式
LargeChange	属性	最大改变值
Layout	属性	图象布局
Length	函数	返回字符串的动态长度
Lines	属性	缺省显示内容
Ln	函数	自然对数函数
Lo	函数	返回参数的低地址位
LoadStr	函数	从应用程序的可执行文件中装载字符资源
LowerCase	函数	将给定的字符串变为小写
Low	函数	返回参数的下限值
Max	属性	最大值
MaxLength	属性	最大长度
Min	属性	最小值
MkDir	命令	建立一子目录
Move	函数	从源到目标复制字节
MultiSelect	属性	允许同时选择几个数据项
Name	属性	控件的名字
New	函数	建立新的动态变量并设置一指针变量指向他
NewStr	函数	在堆栈上分配新的字符串
Now	函数	返回当前的日期和时间
Odd		测试参数是否为奇数
OnActivate	事件	焦点移到窗体上时触发
OnClick	事件	单击窗体空白区域触发
OnDblClick	事件	双击窗体空白区域触发
OnCloseQuery	事件	使用者试图关闭窗体触发
OnClose	事件	窗体关闭后才触发
OnCreate	事件	窗体第一次创建时触发
OnDeactivate	事件	用户切换到另一应用程序触发
OnDragDrop	事件	鼠标拖放操作结束时触发
OnDragOver	事件	有其他控件从他上面移过触发
OnMouseDown	事件	按下鼠标键时触发
OnMouseUp	事件	释放鼠标键时触发
OnMouseMove	事件	移动鼠标时触发
OnHide	事件	隐藏窗体时触发
OnKeyDown	事件	按下键盘某键时触发
OnKeyPress	事件	按下键盘上的单个字符键时触发
OnKeyUp	事件	释放键盘上的某键时触发
OnPaint	事件	窗体上有新部分暴露出来触发
OnResize	事件	重新调整窗体大小触发

OnShow	事件	在窗体实际显示之前瞬间触发
Ord		返回序数类的序数
OutlineStyle	属性	类型
OutOfMemoryError		引起 OutOfMemory 意外
PageIndex	属性	页索引
Pages	属性	页
ParamCount	函数	返回在命令行上传递给程序的参数数量
ParamStr	函数	返回指定的命令行参数
Pen	属性	画刷设置
Pi	函数	返回圆周率 Pi
Picture	属性	显示图象
PictureClosed	属性	设置 Closed 位图
PictureLeaf	属性	设置 Leaf 位图
PictureMinus	属性	设置 Minus 位图
PictureOpen	属性	设置 Open 位图
PicturePlus	属性	设置 Plus 位图
Pos	函数	在字符串中搜索子串
Pred	函数	返回先前的参数
Random	函数	返回一随机函数
Randomize	函数	用一随机数初始化内置的随机数生成器
Read	函数	对有格式的文件，读一文件组件到变量中； 对文本文件，读一个或多个值到一个或多个变量中
Readln	函数	执行 Read 过程，然后跳到文件下一行
ReadOnly	属性	只读属性
ReAllocMem	函数	分配一动态变量
Rename	函数	重命名外部文件
RenameFile	函数	对文件重命名
Reset	函数	打开已有的文件
Rewrite	函数	建立并打开一新的文件
RmDir	函数	删除空的子目录
Round	函数	将实数值舍入为整型值
RunError	函数	停止程序的执行
ScrollBars	属性	滚动条状态
Seek	函数	将文件的当前指针移动到指定的组件上
SeekEof	函数	返回文件的文件结束状态
SeekEoln	函数	返回文件的行结束状态
SelectedColor	属性	选中颜色
SetMemoryManager	函数	设置内存管理器的入口点
SetTextBuf	函数	给文本文件指定 I/O 缓冲区
Shape	属性	显示的形状
ShowException	函数	显示意外消息与地址
Sin	函数	正弦函数
SizeOf	函数	返回参数所占的字节数
SmallChange	属性	最小改变值
Sorted	属性	是否允许排序
Sqr	函数	平方函数
Sqrt	函数	平方根函数
StartMargin	属性	开始边缘

State	属性	控件当前状态
Str	函数	将数值转换为字符串
StrAlloc	函数	给以 NULL 结束的字符串分配最大长度-1 的缓冲区
StrBufSize	函数	返回存储在由 StrAlloc 分配的字符缓冲区的最大字符数
StrCat	函数	将一字符串附加到另一字符串尾并返回合并的字符串
StrComp	函数	比较两个字符串
StrCopy	函数	将一个字符串复制到另一个字符串中
StrDispose	函数	释放堆栈上的字符串
StrECopy	函数	将一字符串复制到另一个字符串 并返回结果字符串尾部的指针
StrEnd	函数	返回指向字符串尾部的指针
Stretch	属性	自动适应控件的大小
StrFmt	函数	格式化一系列的参数
StrIComp	函数	比较两个字符串（不区分大小写）
StringToWideChar	函数	将 ANSI 字符串转换为 UNICODE 字符串
StrLCat	函数	将一字符串中的字符附加到另一字符串尾 并返回合并的字符串
StrLComp	函数	以最大长度比较两个字符串
StrLCopy	函数	将一个字符串中的字符复制到另一个字符串中
StrLen	函数	返回字符串中的字符数
StrLFmt	函数	格式化一系列的参数， 其结果中包含有指向目标缓冲区的指针
StrLIComp	函数	以最大长度比较两个字符串（不区分大小写）
StrLower	函数	将字符串中的字符转换为小写
StrMove	函数	将一个字符串中的字符复制到另一个字符串中
StrNew	函数	在堆栈上分配一个字符串
StrPas	函数	将以 NULL 结束的字符串转换为 PASCAL 类的字符串
StrPCopy	函数	将 PASCAL 类的字符串复制为以 NULL 结束的字符串
StrPLCopy	函数	从 PASCAL 类的最大长度字符串 复制为以 NULL 结束的字符串
StrPos	函数	返回一个字符串在另一个字符串中首次出现指针
StrRScan	函数	返回字符串中最后出现字符的指针
StrScan	函数	返回字符串中出现首字符的指针
StrToDate	函数	将字符串转换为日期格式
StrToDateTime	函数	将字符串转换为日期/时间格式
StrToFloat	函数	将给定的字符串转换为浮点数
StrToInt	函数	将字符串转换为整型
StrToIntDef	函数	将字符串转换为整型或默认值
StrToTime	函数	将字符串转换为时间格式
StrUpper	函数	将字符串中的字符转换为大写
Style	属性	类型选择
Suce	函数	返回后继的参数
Swap	函数	交换参数的高低地址位
Tabs	属性	标记每一项的内容
TabIndex	属性	标记索引
Text	属性	显示的文本
TextToFloat	函数	将字符串（以 NULL 结束的格式）转换为浮点数
Time	函数	返回当前的时间

TimeToStr	函数	将时间格式转换为字符串
Trim	函数	从给定的字符串中删除前导和尾部的空格及控制字符
TrimLeft	函数	从给定的字符串中删除首部的空格及控制字符
TrimRight	函数	从给定的字符串中删除尾部的空格及控制字符
Trunc	函数	将实型值截取为整型值
Truncate	函数	截去当前文件位置后的内容
UnSelectedColor	属性	未选中颜色
UpCase		将字符转换为大写
UpperCase		将给定的字符串变为大写
Val	函数	将字符串转换为整型值
VarArrayCreate	函数	以给定的界限和维数建立变体数组
VarArrayDimCount	函数	返回给定变体的维数
VarArrayHighBound	函数	返回给定变体数组维数的上界
VarArrayLock	函数	锁定给定的变体数组
VarArrayLowBound	函数	返回给定变体数组维数的下界
VarArrayOf	函数	返回指定变体的数组元素
VarArrayRedim	函数	通过改变上限来调整变体的大小
VarArrayUnlock	函数	解锁指定的变体数组
VarAsType	函数	将变体转换为指定的类型
VarCase	函数	将变体转换为指定的类型并保存他
VarClear	函数	清除指定的变体
VarCopy	函数	将指定的变体复制为指定的变体
VarFormDateTime	函数	返回包含日期时间的变体
VarIsArray	函数	测试变体是否为数组
VarIsEmpty	函数	测试变体是否为 UNASSIGNED
VarIsNull	函数	测试变体是否为 NULL
VarToDateTime	函数	将给定的变体转换为日期时间
VarType	函数	将变体转换为指定的类型并保存他
Visible	属性	控件的可见性
WantReturns	属性	为 True 时，按回车键产生一个回车符； 为 False 时，按下 Ctrl+Enter 才产生回车符
Write	命令	对有格式的文件，写一变量到文件组件中； 对文本文件，写一个或多个值到文件中
Writeln	命令	执行 WRITE 过程，然后输出一行结束标志
WideCharLenToString	函数	将 ANSI 字符串转换为 UNICODE 字符串
WideCharLenToStrWar	函数	将 UNICODE 字符串转换为 ANSI 字符串变量
WideCharToString	函数	将 UNICODE 字符串转换为 ANSI 字符串
WideCharToStrVar	函数	将 UNICODE 字符串转换为 ANSI 字符串变量