

1	简介---用 ROBO PRO 软件控制慧鱼模型.....	7
1.1	ROBO Pro 的安装	8
1.2	安装接口板的 USB 驱动程序	9
1.2.1	在 Windows Vista, 7 和 8 系统下安装 USB 驱动.....	10
1.2.2	在 Windows XP 系统下安装 USB 驱动.....	11
1.3	第一步	14
2	编程前的快速硬件测试.....	17
2.1	将接口板和电脑相连.....	18
2.2	控制板的正确设置	19
2.3	错误连接: 未与控制板建立连接?	21
2.4	各部分都工作正常吗 – 控制板测试.....	22
3	级别 1: 第一个控制程序	24
3.1	创制一个新程序	25
3.2	控制程序的模块	26
3.3	插入、移动和修改程序模块.....	27
3.4	连接各程序模块	30
3.5	对首个控制程序的测试.....	32
3.6	其他程序模块	34
3.6.1	时间延迟.....	35
3.6.2	等待输入.....	36
3.6.3	脉冲计数.....	37
3.6.4	循环计数.....	38
3.7	在线和下载操作的差别.....	39
3.8	技巧和诀窍	42
4	级别 2: 用子程序控制.....	43
4.1	你的第一个子程序:	44
4.2	子程序库:	49
4.2.1	使用库:	50
4.2.2	使用你自己的库:	51

4.3	编辑子程序符号	52
4.4	TANGO 控制	53
4.4.1	带脉冲开关的电机控制	55
4.4.2	带编码电机的电机控制	59
4.4.3	TANGO 主程序	61
4.5	TANGO 控制 2	63
4.5.1	ROBO 接口板无线设置	67
4.5.2	TXT 和 TX 控制板的蓝牙设置	70
5	级别 3: 变量, 面板和指令	72
5.1	变量和指令	73
5.2	变量组和多个进程	75
5.3	面板	76
5.4	定时器	80
5.5	子程序指令输入	82
5.6	列表 (阵列)	85
5.7	运算符	87
6	级别 4: 用户定义命令	91
6.1	进程命令	92
6.2	命令过滤器	94
6.3	向子程序发送自定义指令	95
7	控制多个控制板	96
7.1	扩展模块	97
7.2	TXT 控制板, TX 控制板和 ROBO 接口板同时控制	98
7.3	在子程序中接口板的分配	101
7.4	技巧和窍门	102
7.5	改变接口板序列号	103
8	编程模块概览	105
8.1	基本模块 (级别 1)	106
8.1.1	开始	107

8.1.2	结束.....	108
8.1.3	数字量分支.....	109
8.1.4	模拟量分支.....	110
8.1.5	延时.....	111
8.1.6	电机输出.....	112
8.1.7	带编码器电机（级别1）	114
8.1.8	灯输出（级别2）	115
8.1.9	输入等待.....	116
8.1.10	脉冲计数器.....	117
8.1.11	循环计数器.....	119
8.1.12	音乐播放器.....	120
8.2	发射接收模块（级别 2-3）	121
8.2.1	发射器（级别2）	122
8.2.2	接收器（命令接收分支，级别2）	124
8.2.3	接收器（级别3）	126
8.2.4	等待命令（级别4）	127
8.2.5	命令过滤器（级别4）	128
8.2.6	更换指令数据（级别4）	129
8.2.7	I2C 写入（级别4）	130
8.2.8	I2C 读取（级别4）	132
8.3	子程序 I/O （级别 2-3）	134
8.3.1	子程序入口（级别2）	135
8.3.2	子程序出口（级别2）	136
8.3.3	子程序指令输入（级别3）	137
8.3.4	子程序指令输出（级别3）	138
8.4	变量，列表 ... （级别 3）	139
8.4.1	变量（全局）	140
8.4.2	局部变量.....	142
8.4.3	常量.....	143

8.4.4	定时器变量.....	144
8.4.5	列表.....	146
8.5	指令 (级别 3)	149
8.5.1	= (赋值)	150
8.5.2	+ (加)	151
8.5.3	- (减)	152
8.5.4	向右.....	153
8.5.5	向左.....	154
8.5.6	停止.....	155
8.5.7	开启.....	156
8.5.8	关闭.....	157
8.5.9	文本.....	158
8.5.10	添加数值.....	159
8.5.11	删除数值.....	160
8.5.12	交换数值.....	161
8.6	比较, 等待 ... (级别 3).....	162
8.6.1	判断 (带数据输入).....	163
8.6.2	与固定值作比较.....	164
8.6.3	比较.....	165
8.6.4	延时.....	166
8.6.5	等待.....	167
8.6.6	脉冲计数器.....	168
8.7	接口板输入/输出	169
8.7.1	通用输入.....	170
8.7.2	计数输入.....	172
8.7.3	电机到位输入.....	173
8.7.4	电机输出.....	174
8.7.5	灯输出.....	176
8.7.6	面板输入.....	177

8.7.7	面板输出.....	178
8.7.8	摄像头输入.....	179
8.8	运算符.....	180
8.8.1	算数运算符.....	181
8.8.2	比较运算符（关系运算符）.....	182
8.8.3	逻辑运算符.....	183
8.8.4	位运算符.....	184
8.8.5	函数.....	186
8.9	ROBO 接口板.....	188
8.9.1	数字量分支.....	189
8.9.2	模拟分支.....	190
8.9.3	输入等待.....	191
8.9.4	脉冲计数器.....	192
8.9.5	数字量输入.....	193
8.9.6	模拟量输入.....	194
8.9.7	红外线输入.....	196
9	面板模块和面板：概览.....	197
9.1	显示.....	198
9.1.1	仪表.....	199
9.1.2	文本显示.....	200
9.1.3	指示灯.....	202
9.2	控制模块.....	203
9.2.1	按钮.....	204
9.2.2	滑块.....	205
10	绘图功能.....	206
11	摄像头功能.....	208
11.1	摄像头窗口.....	209
11.2	摄像头显示器.....	210
11.3	摄像头探测模块.....	211

11.3.1	颜色探测器.....	212
11.3.2	运动探测器.....	213
11.3.3	线条探测器.....	214
11.3.4	小球探测器.....	216
11.3.5	排除物体.....	218
12	TXT 和 TX 控制板功能.....	219
12.1	安装 ROBO TX 控制板 USB 驱动.....	220
12.2	编程环境（级别 1 及以上）.....	221
12.3	控制板独立编程	222
12.4	程序转换	223
12.5	通用输入，传感器类型和输入模式.....	224
12.6	快速计数输入和扩展电机控制.....	225
12.6.1	带编码器的电机（级别 1）.....	226
12.6.2	级别 3 中的扩展电机控制.....	227
12.7	显示屏	228
13	小数功能.....	230
13.1	对比浮点数	231
13.2	显示浮点数	232
13.3	计算精度	234

1 简介---用 ROBO PRO 软件控制慧鱼模型

你一定曾经问过自己，机器人是如何执行被分配的任务，看上去就像有一只无形的手在操纵它。但不仅仅是真正的机器人，在许多其它涉及到自动化控制技术的领域中同样如此，包括慧鱼机器人。在接下来的章节中，我们将一起来为自动车库门设计一个小的控制程序。这样一来，我们可以知道怎么在 ROBO Pro 软件的帮助下，来解决这类控制问题并进行调试。ROBO Pro 软件非常易于操作。控制程序以及我们即将学到的流程图和数字流程图可以生成图形化的用户界面，这一切几乎用鼠标就可以操作完成。

为了通过电脑来控制你的慧鱼模型，必须要有 ROBOPro 控制软件和一个控制板来将电脑和模型相连。控制板可以传输软件指令，比如控制电机和处理传感器信号。**ROBOTICS TXT 控制板**（货号 522429）、**ROBO TX 控制板**（货号 500995）、ROBO 接口板（货号 93293）和早期的智能接口板（货号 30402）都可以用。你可以选任意一种和 ROBO Pro 软件配套。但是 ROBO Pro 只支持智能接口板的在线控制模式。ROBO Pro 不再支持老式的并行接口板（货号 30520）。

有关这本手册的内容分布：

它分为两个部分。

第一部分，从第一章到第四章，讲述了用 ROBO Pro 编程的基本步骤。这一部分提供了大量的信息和通用的编程背景知识，以及如何使用 ROBO Pro 软件。

第二部分包含第五章至第七章。介绍了以后编程所需的一些功能模块。

第八章之后是参考部分，所以在你阅读完第一部分且熟悉了 ROBO Pro 的操作之后，需要非常明确的信息，这里你可以找到各个单独编程模块的详细解释。

如果你对 ROBO Pro 软件非常熟悉，只是想了解 ROBO TXT 控制板的新功能，你可以阅读[手册 11 章：摄像头功能](#)。

你一定已经非常渴望知道如何来用 ROBO Pro 来对你的慧鱼机器人进行编程吧。OK，我们开始吧！

1.1 ROBO Pro 的安装

安装 ROBO Pro 的系统要求:

- 微软视窗操作系统 Windows XP, Vista, 7 or 8。
- 1 个空闲的 USB 接口，用以连接 ROBOTICS TXT 控制板，ROBO TX 控制板连接，或 ROBO 接口板。
- 安装有 PDF 阅读器（如 Adobe Reader），用以阅览 ROBOTICS TXT 控制板使用手册，和 ROBO Pro 软件附带的 ROBOTICS 系列组合包的电子活动手册。

首先，启动计算机登陆操作系统。控制板只有在软件正确安装后才能电脑相连。将安装光盘插入光驱，安装程序就自动启动了。

- 在安装程序第一个的欢迎窗口中，你只需按一下**下一步(NEXT)**按钮。
- 第二个窗口是**重要提示**，包括重要的程序安装和程序本身更新提示。这里也只要按 **NEXT** 按钮。
- 第三个窗口是**许可协议**，显示 ROBO Pro 的许可契约。你必须按 **YES** 接受协议并按**下一步(NEXT)**进入下一个窗口。
- 下一个窗口是**用户详细资料**，请输入你的名字等信息。
- 下一个窗口是**安装类型**，允许你在**快速安装**和**自定义安装**中选择。在自定义安装中，你可以选择单个组件来安装。如果你是在旧版本的 ROBO Pro 基础上安装新版本的 ROBO Pro，而且你已经修改了旧版本的范例程序，你可以选择不安装范例程序。如果你不这么做，你已经修改过的旧版本范例程序会在**没有提示的情况下被自动覆盖**。如果你选择自定义安装并按**下一步(NEXT)**，会出现一个新的选择组件窗口。
- 在**安装目标目录**窗口，允许你选择将 ROBO Pro 安装到的目标文件夹或者目录。默认路径是 **C:\Programs\ROBO Pro**。当然，你可以选择其他的路径。
- 当你在最后一个窗口，按下 **Finish** 按钮，安装就完成了。安装一旦结束（一般需要等几秒钟），程序会提示安装成功。如果安装有问题，会有错误信息出现，帮助你解决安装问题。

1.2 安装接口板的 USB 驱动程序

当 ROBOTICS TXT 控制板, ROBO TX 控制板或 ROBO 接口板通过 USB 连接到电脑之后, USB 驱动会自动安装完成。

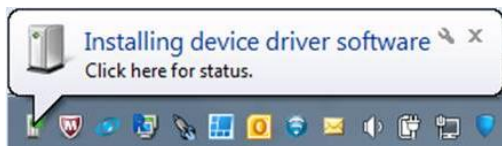
安装 USB 驱动的重要提示:

USB 驱动程序需要有系统管理员的权限才可以安装。你必须请你的系统管理员来安装驱动程序或者不安装这个驱动程序, 否则安装程序会提示你没有安装 USB 驱动程序的权限。

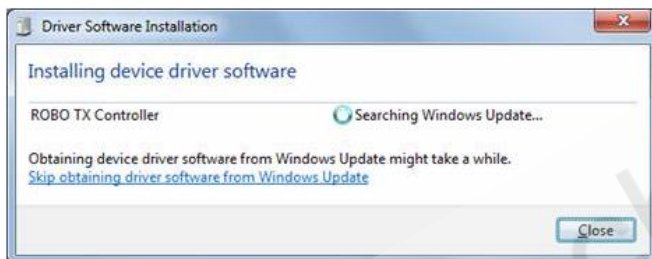
为了安装 USB 驱动, 首先需要通过一根 USB 数据线将 ROBOTICS TXT 控制板, ROBO TX 控制板或 ROBO 接口板连接到电脑, 并给控制板供电。Windows 操作系统会自动识别到连接的新设备, 根据操作系统不同, 显示的信息会有不同。

1.2.1 在 Windows Vista, 7 和 8 系统下安装 USB 驱动

首先会出现以下信息：



如果用鼠标左键单击蓝色信息部分，会出现以下对话框：



自动寻找驱动(Searching Windows Update)会花费一些时间。由于安装驱动所需的文件已经包含在已安装的 ROBO Pro 当中，驱动文件就无需下载。不要点击“关闭(Close)”按钮，否则将会出现错误。一旦 USB 驱动安装完成，会出现以下信息：



上图表明驱动已安装完成。注意：上图是在 Windows 7 系统下显示的信息，Windows Vista 和 8 可能略有不同。

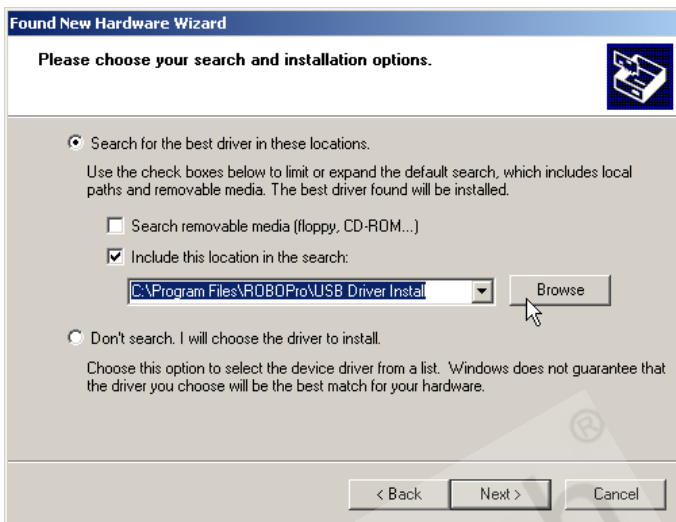
1.2.2 在 Windows XP 系统下安装 USB 驱动

在 Windows XP 系统下的安装过程与上述系统环境下有很大不同，具体步骤如下：



你必须选择**从列表或特定位置安装**(Install from a list or specific location)，再按下**下一步**(Next)。

在下一个窗口，你不要选择**从移动设备中搜索**(Search removable media)选项，选择在**以下位置搜索**(Include this location in the search)选项。单击**浏览**(Browse)，选择 ROBO Pro 安装目录（通常是 C:\Programm Files\ROBOPro\）下的 **USB Driver Installation** 子目录。ROBOTICS TXT 控制板选择 TXTControllerontrroller，ROBO TX 控制板选择 TXControllerontrroller，ROBO 接口板选择 ROBOInterface。



在 Windows XP 平台上，在按下一步(Next)后，你可能会看到一下信息。



这个 USB 驱动程序正经微软测试。一旦测试完成，微软会更新驱动程序，这个信息就不会出现了。需要安装这个驱动程序，请按**下仍然安装(Continue Anyway)**。

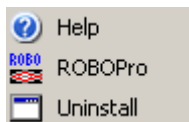
最后会出现如下信息:



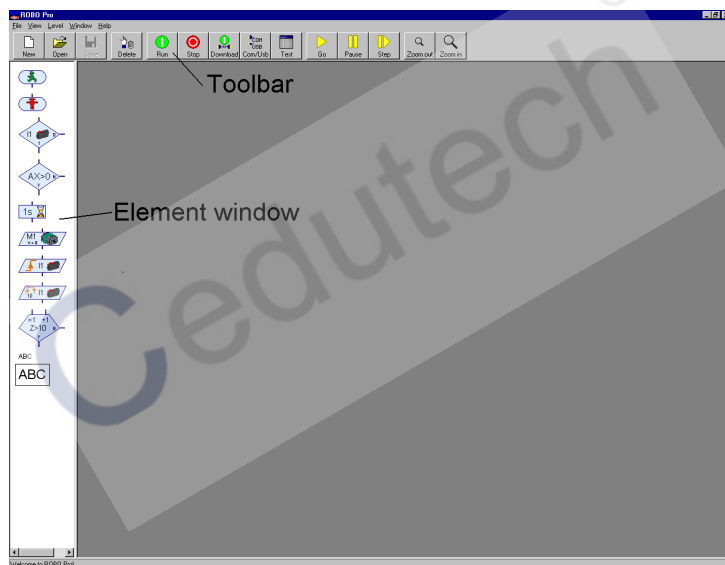
单击**完成(Finish)**，USB 的驱动程序就安装完毕了。

1.3 第一步

好奇吗？那么启动 ROBO Pro 软件吧。只要点击任务栏中的**开始按钮**，然后选择“**程序**”或者“**所有程序**”和 **ROBO Pro**。在开始菜单中，可以找到如下几个选项：



选择“**卸载(Uninstall)**”选项可以方便地卸载 ROBO Pro 软件。选择“**Help**”选项可以打开 ROBO Pro 的帮助文件，而选择“**ROBO Pro**”可以打开 ROBO Pro 程序。现在选择“**ROBO Pro**”启动程序。

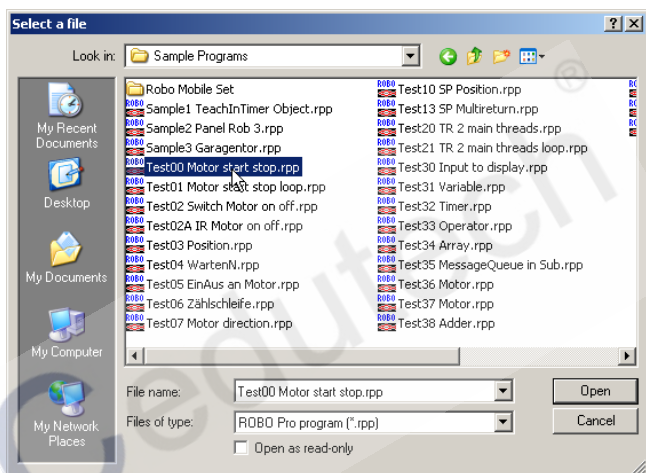


窗口中有一个菜单栏和工具栏，上面有各种操作按钮，左面的窗口里还有各种不同的编程模块。如果在左边出现了两个层叠的窗口，那么 ROBO Pro 没有设定在“**级别 1**”。为了让 ROBO Pro 功能适应你知识的增长，可以将 ROBO Pro 设定在级别 1 的初学者和级别 5 的专家级之间。打开“**级别(Level)**”菜单看是否有标识为**级别 1：初学者(Level 1: Beginners)**。如果不是，请切换到级别 1。

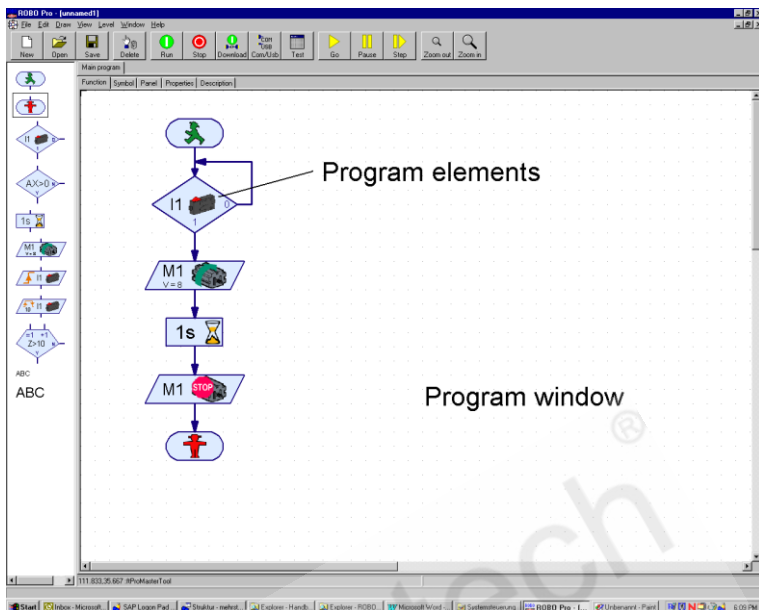


ROBO Pro 默认使用 ROBOTICS TXT 控制板或 ROBO TX 控制板，可以在工具栏上看到 ROBO TX/TXT 按钮。在 [12.2 章的编程环境小节](#)中，您将学习如何切换到 ROBO 接口板的编程环境和注意事项。

现在你可以创建一个新程序，也可以打开一个已经存在的程序文件。我们在第三章才会去写我们的第一个控制程序，之前还不打算创建新程序。为了更好熟悉全新的用户界面，我们来打开一个现成的范例程序。这样，你可以点击**文件(File)**菜单中的**打开(Open)**选项，或者用工具栏中的**打开(Open)**按钮。范例程序可以在文件夹 **C:\Program Files\ROBO Pro\Sample programs** 中找到。



打开文件 **Test00 Motor start Stopp.rpp**:



这里可以看到一个简单的 ROBO Pro 程序的外观。编程时，将模块窗口中的编程模块在编程窗口中组建成控制程序流程图。然后，在用接口板进行测试之前，可以对已完成的流程图进行检查。但不要太快了：我们应该在以下的几章中一步步地学习编程！你已经对用户程序有第一印象了吧，你可以用**文件(File)**菜单中的**关闭(Close)**指令关闭程序文件。对于是否要保存文件，可以回答**否(No)**。

2 编程前的快速硬件测试

很明显，必须先将控制板和电脑相连，以便稍后可以测试我们要新建的程序。但是，根据所连接的控制板（ROBOTICS TXT 控制板，ROBO TX 控制板，ROBO 接口板），必须进行适当的软件设置和连接的测试。在接下来章节里就将进行这项工作。

Cedutech®

2.1 将接口板和电脑相连

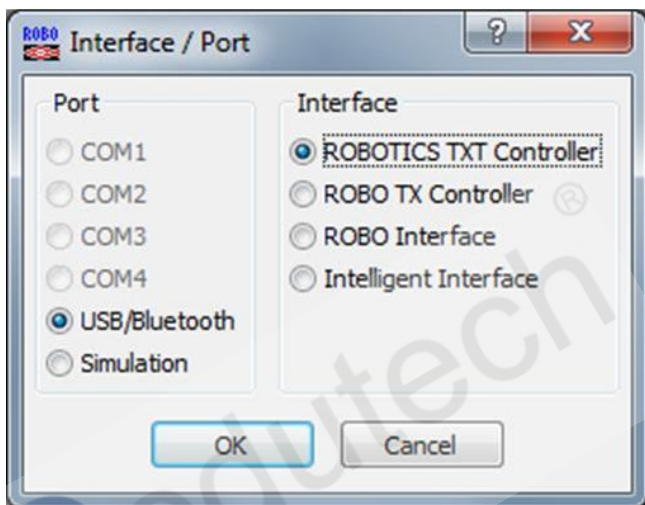
这应该不是个大问题。将随控制板所带的 USB 数据线一端接到接口板，另一端接到电脑：

这些端口通常可以在计算机机箱的后部找到。各个连接端口的准确位置应该在你的电脑用户手册中有准确的描述；请查阅。USB 端口也经常可以在你的电脑前部找到。不要忘了给控制板供电（开关电源或者电池）。各个控制板的连接在相应的用户手册中有详细描述。

Cedutech®

2.2 控制板的正确设置

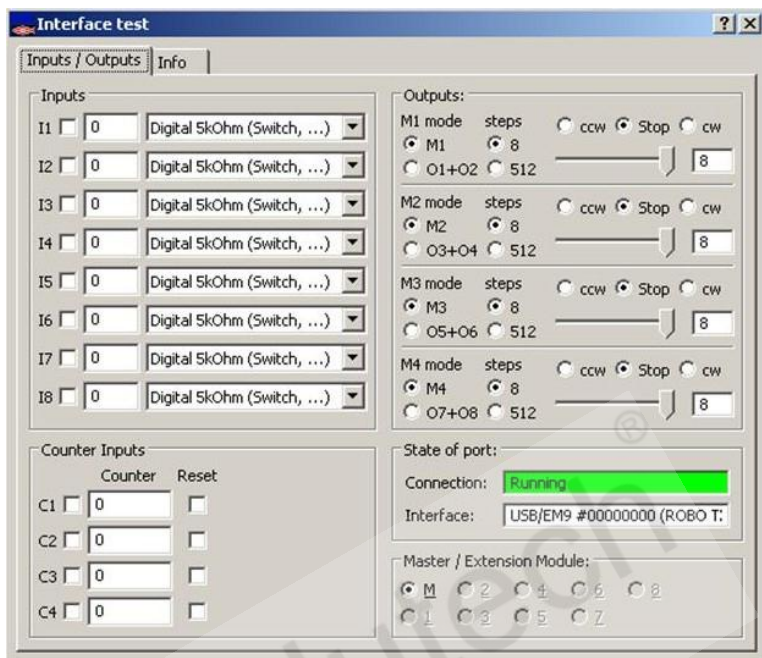
为了使控制板和电脑间的连接工作正常，ROBO Pro 必须对当前使用的控制板进行设置。具体来说，用开始菜单中的 **Programs** 或者 **All programs** 下的 **ROBO Pro** 来启动 ROBO Pro 程序，然后单击工具栏中的 **COM/USB**。出现如下的窗口：



这里你可以选择端口和控制板的类型。

一旦选定了适当的设置，单击“OK”，关闭窗口。然后，可以单击工具栏中的“**Test**”，打开控制板测试窗口。





其中显示了控制板有效的输入和输出。窗口下方的绿条显示了电脑和控制板的连接状态。

- **Connection:Running** 表明已与控制板正确连接。
- **Connection:Stopped** 表明电脑和控制板没有建立正确连接，状态条的显示为红色。

为了更换控制板或者改变连接设置，你必须先关闭测试窗口（点击右上方的 **X**），然后通过工具栏中的 **COM/USB**，选择如前所述的其它端口或者其它接口板类型。

如果你已经在电脑和控制板之间建立了连接，而且出现了绿色的状态条，那么你可以放心地跳过下一节的内容。

如果没能建立连接，也许下一节的一些提示可以帮你解决问题。

2.3 错误连接: 未与控制板建立连接?

尽管你已经正确地设置了端口(见上一节),但还是得到了“Stopped”的信息,你应该在以下几点中寻找原因。另外,你还可以从电脑专家那里寻求建议:

- **电源供电:**

接口板用了适当的电源吗? 如果你随意取用了电池或者是可充电的电池作电源,问题可能在于电源的电压不足了。如果电池的电压跌到 6V 以下, ROBO TX 控制板的处理器就停止工作了。这种情况下,显示屏不会显示任何信息。如果电压过低,你必须更换合适的电池,或者给电池充电。如果可能的话,最好使用直流稳压电源来测试控制板。

- **USB 驱动是否正确安装?**

你可以在设备管理器的 COM 和 LPT 连接下找到全部通过 USB 连接的 ROBO TX 控制板。如果没有出现,重新安装 USB 驱动,如果出现错误,卸载驱动(鼠标右键点击卸载)并再次安装。如果 USB 驱动无法自动安装,你可以手动安装。你可以在 ROBO Pro 的文件夹下找到不同设备的驱动文件(默认路径为 C:\Program Files(x86)\ROBOPro\USB driver installation)。另外手动安装 USB 驱动的可以查看 [1.2 安装接口板的 USB 驱动程序](#),或者上官网查询 www.fischertechnik.de-Downlads-ROBOTICS

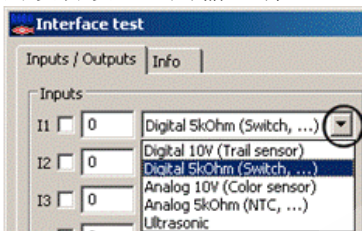
- 是否和其它设备驱动器(外置调制解调器)在同一个端口上冲突? 需要把此驱动器置为无效(见微软或设备手册)。
- 只发生在 Windows NT/2000/XP 和智能接口板配合的系统中: 如果一块早先的智能接口板已经和电脑相连,电脑在上电启动的时候, Windows NT 会将接口板切换到下载模式。为了重新和电脑建立连接,你仅仅需要暂时中断接口板的电源。这个问题对于新的 ROBO 接口板不存在。
- 如果你还是无法和控制板建立连接,那有可能是控制板或者连接电缆出错了。这种情况下,你可以咨询慧鱼服务部门。

2.4 各部分都工作正常吗 – 控制板测试

一旦连接正确建立了，我们就可以通过控制板测试窗口来测试控制板和与它相连的模型。如前所提及的，测试窗口显示了控制板的各种输入和输出：

● 通用输入 I1—I8

I1—I8 是 ROBOTICS TXT 控制板和 ROBO TX 控制板的通用输入端。这里可以连接各种传感器，包括数字量和模拟量。你可以根据要连接的设备设置通用输入端口。



数字量输入只有两种状态 0 和 1，或者 Y 和 N。默认情况下，所有通用输入都被设置为数字量 5kOhm 模式。开关（迷你微动开关），光电晶体管（光电传感器）或者干簧管（磁传感器）可以作为数字量输入来连接。

你可以将一个迷你微动开关（货号 37783）接到控制板上，比如 I1，来检查这些端口的功能（用开关上的触点 1 和 3）。一按下开关，I1 的显示将出现一个检查标志。如果你连了开关的另一种方式（触点 1 和 2），当你按下开关的时候检查标志就消失了。

- **数字量 10V** 模式用于红外轨迹传感器。
- **模拟量 10V** 模式可以用于测量 0 到 10V 的电压，例如电池的电压，数值单位为 mV（毫伏）。
- **模拟量 5kOhm** 模式可以用于 NTC 电阻测量温度，光敏电阻测量光强度，数值单位为 Ohm（欧姆）。
- **距离模式**用于超声波距离传感器（对于 ROBOTICS TXT 传感器和 ROBO TX 传感器，只有 3 线的超声波距离传感器才能使用，货号为 133009）。
- **计数输入 C1—C4**

这些输入端口可以用以测量快速脉冲，频率可达每秒 1000 脉冲。同样这些端口也可以作为数字量输入端（不适用于轨迹传感器），如果将微动开关连接到这些端口，每次按动按钮（1 个脉冲）将会增加 1 个计数。这可以让机器人跑动特定的距离。

- **电机输出 M1—M4**

M1—M4 是控制板的输出，这里可以连接所谓的执行器，可以是电机、电磁铁或者灯。这 4 路电机输出可以改变方向和速度。速度用滑块控制，可以选择用粗略的 8 级调速或精确的 512 级调速。在级别 1 和级别 2 下，只能使用 8 级调速，从级别 3 开始，可以使用 512 级调速，速度数值可以在滑块旁显示。如果你要测试输出，可以将一个电机接到输出端，比如 M1。

- **灯输出 O1—O8**

每个电机输出也可以用作一对单个的输出。这些输出不仅可以用作灯的控制，也可以用作单向电机的控制(比如传送带电机)。如果你要测试其中一个输出，可以将一个灯接到输出，比如 O1。可以将灯的另一个接到控制板的接地插孔(L)。

- **扩展板**

额外同种类型的控制板或扩展模块可以连接到控制板（请查看具体设备的使用手册）。这些选项可以选择在测试窗口下测试的是哪一个设备。

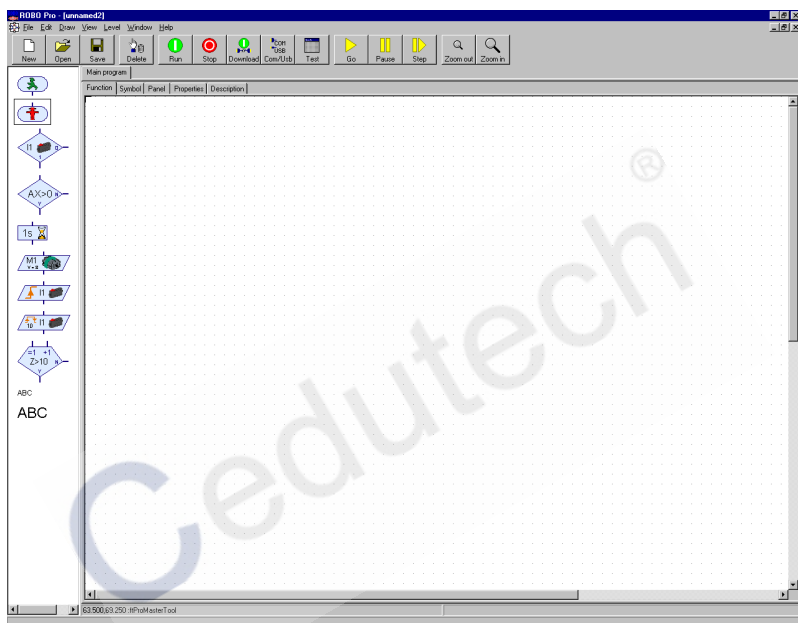
3 级别 1：第一个控制程序

测试完了硬件，即控制板和连接其上的开关与电机（第一章中提到），我们将开始着手编程。但“编程”到底是什么意思呢？那么请想象一下，例如，一个机器人与我们的控制板连接。但这个机器人因为非常笨，无法自己独立地工作。幸好，我们比它要聪明一点。因此我们可以告诉它究竟该做些什么以及怎么做。还记得在上一章中我们用鼠标左键放置在电机输出 **M1** 上，发生了什么吗？是的，我们启动了电机。如果这个电机用来驱动机器人的夹爪，那我们只需要告诉机器人：“抓住那个物体！”但现在我们不想每一步都自己动手做，我们希望让机器人自动地完成这一切。为了达到这个目的，我们必须首先储存每一个将被执行的步骤，只有这样，机器人才能逐个地完成这些步骤。即我们必须创建一个可以控制机器人的程序，专业术语称之为“控制程序”。

3.1 创制一个新程序



ROBO PRO 软件为我们设计控制程序并借助连接的控制板进行测试，提供了一个很好的平台。别担心，我们不会立即开始给机器人编程。我们首先要进行一些简单的控制任务。因此我们必须创建一个新的程序。在工具栏中，你可以看到“新建(New)”。如果你把鼠标左键点击它，即可建立一个新程序。



现在，你将看到一个白色大区域，你将在这里面编写第一个程序。如果你在左边的边缘区域内看到两个层叠的窗口，请切换到“级别(Level)”菜单中的“级别 1: 初学者(Level1:Beginners)”。

3.2 控制程序的模块

现在我们可以开始创建我们第一个控制程序了。我们将基于一个具体的例子：

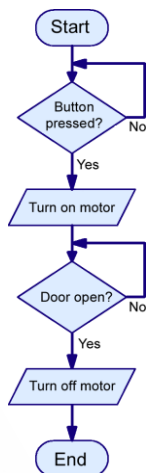
功能描述：

请想象一扇可以自动打开的车库大门，可能你家里就有一扇！你开着车到了车库门口，然后按了一下传达器的按钮，门便由一个电机牵引着打开了。这台电机一定要保持运转，直到门完全打开。

用文字实在是很难形象地描述一个控制程序，因此我们用“**流程图**”来帮助描述一系列将被执行的动作以及完成这些动作所需的条件。在我们的控制系统中，动作“启动电机”的条件是按下按钮。这些流程图读起来很容易：一步步按着箭头的顺序就可以了！同时也展示了控制系统的工作过程——每一个步骤都只能沿着箭头所指的路径完成，而不是任何其他路径。这样可以省下很多麻烦，不是吗？

我们可以利用 ROBO PRO 软件精确地画出这张流程图，并依此为连接着的硬件（控制板，电机，开关等）创建**控制程序**。其它的任务都由软件完成，这与操纵大型工厂设备的原理相同。因此让我们集中精力来完成流程图创建吧。

你将各种程序模块连到一起形成了流程图。又一个新的概念？别担心！在 ROBO PRO 软件中，把放在一起形成流程图的各个模块称之为程序模块。“启动电机”这一动作意味着：控制板应该事实上启动连接在其上的电机！你可以在左手边的模块窗口中找到有效的各种程序模块。



3.3 插入、移动和修改程序模块

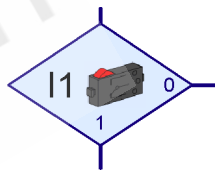
现在的任务是根据模块窗口中的程序模块创建一个车库大门控制系统的流程图。所有可被利用的程序模块都可从模块窗口中取得，并插入到程序窗口中。

插入程序模块

把鼠标移动到想使用的程序模块的符号上，并单击左键。然后把鼠标移动到程序窗口内（即那块白色的大区域），再单击一次。你也可以通过按住鼠标键把程序模块拖入程序窗口。程序总是起始于一个“开始”模块。它是一个有着正在行走的小绿人的跑道形状按钮。这最方便来尝试一次插入操作：在模块窗口中直接用鼠标左键单击“开始”模块，把鼠标移到程序窗口中，再单击一次左键。



程序流程图中的下一个模块可以查询输入，并按照其不同状态进入不同的分支。在模块窗口中，用鼠标点击在正确的模块上，并将其移动到刚才插入的“开始”模块下。如果“分支”模块的上部输入端在“开始”模块下部输出端的下方一两个格子，那么程序窗口中会出现一条连接线。如果再次单击左键，则“分支”模块会被插入，并自动与“开始”模块连接。



移动程序模块和组

可以通过按住鼠标左键，将一个已插入的程序模块移动到理想的位置。如果你想将多个模块同时移动，你可以首先按住鼠标，沿着这些模块的外围画出一个框。具体做法是：在空白区域单击左键，并按住左键不放，用鼠标画出一个包含了所需模块的矩形区域。在此矩形区域中的模块将会显示为有红色的边框。你只要用鼠标左键移动这些红色模块之中的一个，所有的红色模块都被同时移动。你还可以用左键单击单个的模块，同时按住 **shift** 键，来同时选中它们。如果你将左键在空白区域单击，所有的红色标记的模块全部都会再次回到原来的正常状态。

复制程序模块和组

有两种方法复制程序模块和组。一种方法和移动模块差不多，只是在移动前必须先按住键盘上的 **CTRL** 键并且不放，直到移到了指定位置。这样，模块并未被移动，而是被复制了。但是，你只能用这种方法将模块复制到同一个程序中。如果你希望将模块从一个程序复制到另一个程

序中，你可以使用窗口中的剪贴板。首先用前一部分中描述的移动模块的方法，选中一些模块。然后同时按下键盘上的 **CTRL** 和 **C** 键，或者在编辑菜单中选择“复制”，于是所有的已选模块都会被复制到窗口中的剪贴板上。接着你可以切换到另一个程序中，并通过同时按下键盘上的 **CTRL** 和 **V** 键，或者在编辑菜单中选择“粘贴”，再次在新程序中插入模块。一旦模块被复制，你可以无数次地粘贴它们。如果你想将模块从一个程序移动到另一个，你可以在第一步时，同时按下键盘上的 **CTRL** 和 **X** 键，或者在编辑菜单中选择“剪切”，而非 **CTRL** 和 **C** 键，或“复制”。

删除模块和撤销功能

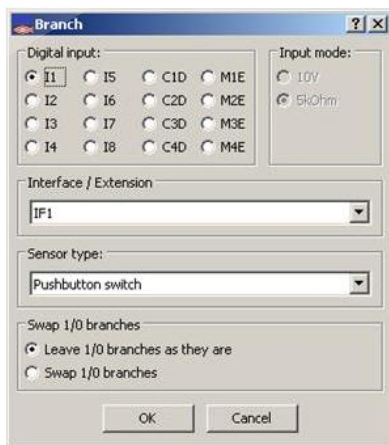
删除模块很容易。你可以通过按下键盘上的“delete”键(Del)，删除所有标记为红色的模块。同样也可以用“删除”功能删除单个模块。具体做法是，首先在工具栏中点击如左图所示的按钮，然后在要删除的模块上点击一下。现在就试试吧。然后，你可以重新插入被删除的模块，也可以利用“编辑”菜单中的“撤销”功能恢复已被删除的模块。使用这个菜单项，你可以撤销任何对程序所作的改动。



编辑程序模块的性能

如果你用鼠标右键点击程序窗口的程序模块，会出现一个对话框，你可以在里面改变模块的各种属性。“分支”模块的属性窗口如右图所示。

- 在 **I1** 至 **I8** 按钮的选项中，你可以选择所要查询的控制板的输入端。输入端 **C1D** 到 **C4D** 对应相应作为数字量端口的计数输入端，**M1E** 到 **M4E** 会在稍后讲解。
- **C1D** 到 **C4D** 输入和 **M1E** 到 **M4E** 的详细介绍请查看 [8.1.3 数字量分支模块](#)。
- 接口板 / 扩展板选项请查看 [第 7 章扩展模块和多个接口板的控制](#)。
- 在传感器类型(Sensor type)一栏中，你可以选择与输入端相连的传感器。数字量输入最常用的是微动开关，也经常使用光电传感器或干簧管开关。自动选择传感器需要使用 ROBOTICS TXT 控制板和 ROBO TX 控制板的 **I1-I8** 通用输入。

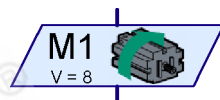


- 在**改变 1/0 位置(Swap 1/0 connection)**一栏中,你可以交换分支出口 1 与分支出口 0 的位置。通常出口 1 在下方,出口 0 在右边。但有时让出口 1 在右边更实用。选中 **Swap 1/0 connection**, 则一旦选择 **OK** 并关闭窗口, 连接 1 与 0 就会立即更换位置。

小贴士: 如果使用迷你开关的一对常开触点, 1 端与 3 端, 则一旦按下开关, 程序将连入分支 1, 而非分支 0。

如果使用迷你开关的一对常闭触点, 1 端与 2 端, 则一旦按下开关, 程序将连入分支 0, 而非分支 1。

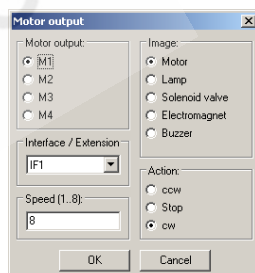
我们的车库门控制系统中下一个模块是“电机”模块。用和插入前两个模块一样的方法将“电机”模块插入“分支”模块下。最好插在一个可以使其自动与以上模块连接的位置。



通过“电机”模块,我们可以控制电机,电灯或者电磁铁。同样,你也可以通过右击模块来打开电机模块的属性窗口。

- 你可以通过选择 **M1** 至 **M4**, 来选择所要控制的控制板输出端口。
- 在**类型**一栏中, 你可以选择代表连接到输出端的慧鱼元件的图示。
- **接口板 / 扩展板**选项请查看[第 7 章扩展模块和多个接口板的控制](#)。
- 在**动作状态**一栏中, 你可以选择输出动作类型。可以让电机向左转(逆时针), 向右转(顺时针)或者停止电机。同样也可以控制一盏灯。
- 在**速度强度(Speed/Intensity)**一栏中, 你可以设定电机运转的速度或者灯的亮度。可能的数值为 1 至 8。

在我们的流程图中, 我们应把参数置为电机 **M1** 在**速度 8** 左转。

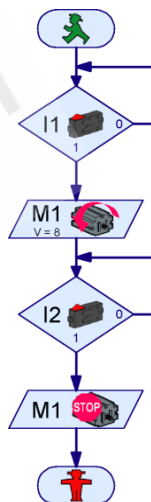
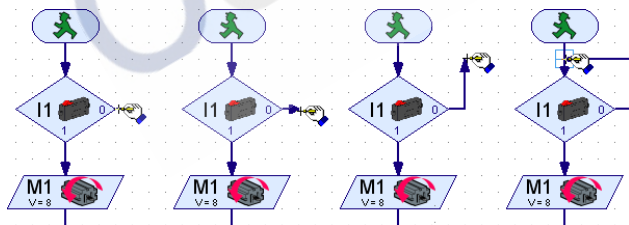


3.4 连接各程序模块

现在，你已经知道了如何把模块插入控制程序，我们就可以继续来完成控制程序的编制。回想一下车库门控制系统的功能描述：还有遗漏的吗？是的，我们可以通过按按钮来启动电机，但一旦门打开，电机应该能自动关闭！在实践中，这是由限位开关来实现的。这个传感器安装在门上，直到门完全打开的一刻，传感器动作。与启动电机时一样，这个信号可以用来关闭电机。我们可以再次使用“分支”模块来查询限位开关的状态。

因此，在程序中插入另一个判断模块，用来查询限位开关 I2 的状态。别忘了右击模块，对输入 I2 进行设置。一旦车库门完全打开，并且压住了限位开关，电机就应该停下来。通过使用“电机”模块就可以做到这一点，和我们启动电机用的是同一个模块。如果你鼠标右键点击模块，可以通过改变模块的功能来使电机停止。程序在“停止”模块处结束。你的程序应该与右图基本相同。

如果你放置的模块相互间相隔仅一两格子，则大多数的进口与出口都将由程序流程来连接。但两个“分支”模块的 No (N) 出口还未被连接。只要输入 I1 的按钮未被按下，程序应退回并重新查询开关状态。可以通过相继在下图所示处点击鼠标，来连接这条线。



小贴士：如果线没有被正确连接到一个接点或另一条线，将会在箭头处出现绿色矩形。在此情况下，你应该通过移动或删除及重画线条来重新建立连接。否则，程序运行到了这一点就不会再运行下去。

删除程序流程线

删除程序流程线和删除程序模块的方法一样。鼠标左键单击这条线，使得它显示为红色。然后按下键盘上的删除(Del)键来删除这条线。如果

同时按住 **shift** 键，然后连续点击那些线，你也可以选中多根线。除此以外，你还可以通过框起这些线，来选中它们，然后再按下 **Del** 键一下子删除所有红色的线。

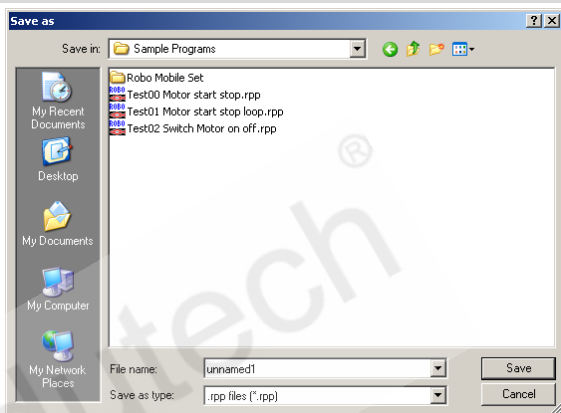
Cedutech®

3.5 对首个控制程序的测试

为了测试我们的首个控制程序，你应该建立一个小型模型。为了达到这一点，在控制板上将开关连接到 I1 与 I2，同时接一个电机到 M1 就可以了。

注意：如何将控制板连接到计算机以及如何建立控制板设置已在前几章中讲到，你可以参见前面几章。

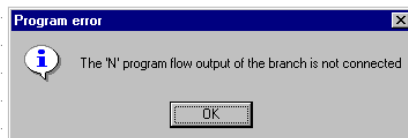
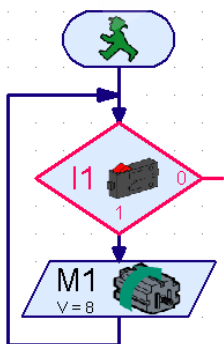
在测试程序前，你应该在你的电脑硬盘上保存程序文档。点击 **File(文件)** 菜单中的指令 **Save as(保存为)** 以下窗口会出现：



在 “Save in”(保存位置) 中，选择你想要保存的目录。在 “Filename”(文件名) 中，输入一个还未被使用的名字，例如：

GARAGE DOOR (车库门) (注意：保存的文件名最好不要出现汉字)，然后用鼠标左键单击 “Save”(保存) 来确认。

为了测试这个程序，应按下工具栏中的开始键 (见左图)。首先，ROBO Pro 会测试是否所有程序模块都被正常连接。如果由某个模块没有适当连接或出现一些顺序错误，会标示为红色，描述错误的信息会出现。例如：如果你忘了连接一个程序分支的 No(N) 出口，以下信息会出现：

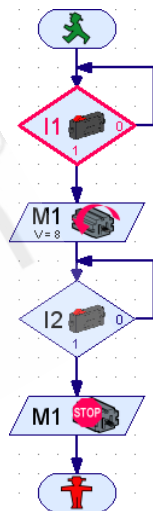


如果你已经接受了一条错误信息，你必须首先纠正其中指出的错误。否则，程序无法启动。

注意：你可以在 [3.7 章节](#) 找到这种操作模式和“下载操作”模式的详尽解释。

第一个“分支”模块将被标示为红色。这表示程序正在模块处等待某一事件的发生，即按钮 I1 的按下，因为这样可以使大门打开来。只要在输入 I1 处的开关未被按下，程序转到 **No(N)** 出口并重新回到分支的开始处。现在按下与接口板的输入 I1 连接的开关。这样就满足了继续下去的条件，于是电机就启动了。下一步，程序等待着在输入 I2 上的限位开关被按下。一旦你按下接在 I2 端的限位开关，程序的分支将会转到第二个电机模块，使电机再一次停止。最终，程序将到达程序终点处。此时会出现一条信息，告知程序已结束。

一切都畅通无阻吗？恭喜了！这意味着你已经创建并测试了你的首个控制程序。如果程序没有正常运作，别泄气，再重新仔细检查一遍：一定在哪里还隐藏着一个错误。没有一个程序员是不犯错误的，同时，犯错是学习的最好方法，所以加油吧！



3.6 其他程序模块

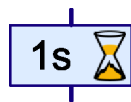
如果你已经将首个控制程序在真正的车库门模型上作了实验，那么现在门应该可以打开了。那我们能否再将其关上呢？当然可以。我们可以再次通过按按钮来启动电机！但我们想用其它的方法，并且学习一种新的程序模块。为了达到这一点，你首先应用一个新名字保存程序（我们以后还会用到当前的流程图）使用 **File（文件）** 菜单中的 **Save as（另存为）**，并输入一个未被用过的文件名。

Cedutech®

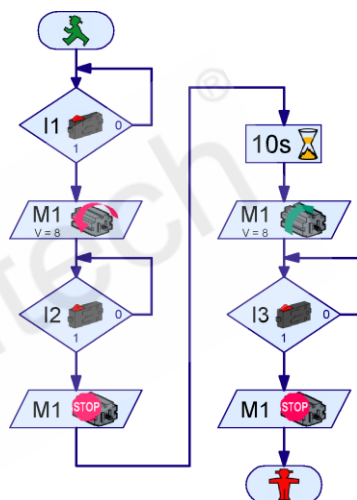
3.6.1 时间延迟

在我们可以扩展流程图之前，必须删除在“关闭电动机”和“程序停止”之间的连接，并且将停止模块向下移。

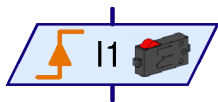
现在，你可以在这两个模块之间插入新的程序模块。假设车库大门将在 10 秒后自动关闭。为了达到这一点，你可以使用**时间延迟**程序模块（见右图）。你可以通过鼠标右键单击模块，在一定的时间范围内，设定自己需要的等待时间。这里，输入 10 秒为理想的时间延迟。为了关上车库大门，电机向另一个方向，即顺时针运转。并且电机在另一个限位开关 I3 压住时关闭。



最终的流程图看起来应该大致如右图所示。为了演示，新的程序模块被搬到了右边。一旦流程图中没有错误，你就可以按下 **Start（开始）** 按钮，来测试扩展了的车库门控制系统。按下 I1 处的按钮，电机启动。并在 I2 处的限位开关压下时关闭。这就是如何打开车库门。现在经时间延迟模块延时了 10 秒，是我们设定的。然后，电机开始反向运转，直到在 I3 处的限位被压下，电机停止运转。你可以试着改变一下延迟时间。



3.6.2 等待输入



除了时间延迟模块，还有另外的两个模块，用来等待一些使程序继续运行的东西。如左图所示的**等待输入**模块，等待控制板的某个输入由一种特定的方式改变为一种特定的状态。这个模块

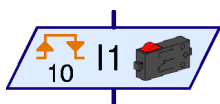


共有五种不同的形式。

符号					
等待	输入=1 (闭合)	输入=0 (打开)	跳变 0-1 (打开到闭合)	跳变 1-0 (闭合到打开)	任一跳变 (1-0 or 0-1)
用“分支”模块实现相同功能					

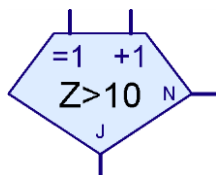
“等待输入”模块也可以由“分支”模块的组合来代替，但是**等待**导入模块更简单，更容易理解。

3.6.3 脉冲计数

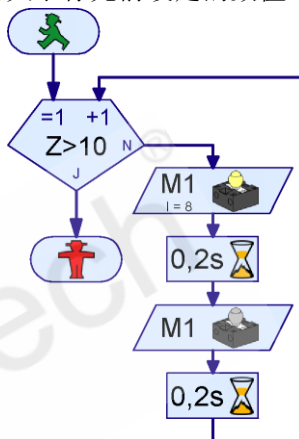


很多慧鱼机器人模型都使用脉冲齿轮。这些齿轮每旋转一圈会触动四次开关。有了这些脉冲齿轮，你可以以一个精确的转数来驱动电机，而不是根据给定的时间。为了达到这一点，你需要计算齿轮的某个输入处的脉冲数。**脉冲计数模块**（见左图）就是用来等待用户定义的脉冲数。对于这种模块，同样地，你可以设定所计脉冲为 **0-1**、**1-0** 或者两者皆可的。脉冲齿轮通常等待双向的变化，这样用一个四齿脉冲轮达到了每转 **8** 个脉冲的精度。

3.6.4 循环计数



有了循环计数模块，你可以十分简单地将程序中的特定部分多次运行。例如右图所示的程序，把接到 M1 处的灯开关 10 次。循环计数模块有一个内部计数器。如果循环计数通过 **=1** 入口进入，则计数器被置为 1。如果循环计数通过 **+1** 入口进入，则计数器加上 1。根据计数器显示数值是否大于你先前设定的数值，循环计数分支将转到 **Yes(Y)** 或 **No(N)** 出口。因此，只有当循环次数与你先前设定的数值相等时，循环计数分支才会转到 **Yes** 出口。从另一方面来说，如果需要进一步的循环，循环计数分支将会转到 **No** 出口。作为一种判断模块，你也可以通过属性窗口将 **Yes** 与 **No** 出口互换。



3.7 在线和下载操作的差别



至此，我们已经用被称之为**在线操作**的方式测试了控制程序。这样，你可以在屏幕上跟踪程序的进程，因为当前活动的模块在屏幕被标示成红色。你可以用在线方式来帮助理解程序或者找出程序中的错误。



在线方式下，你还可以通过按 **Pause（暂停）** 按钮来停止程序并继续执行程序。这非常实用，因为它可以使你在不停止程序的情况下，得到一些有关你的模型的数据和资料。如果你正试图理解程序运作的原理，暂停按钮十分有用。



有了 **Step** 按钮，你可以一个模块一个模块地分步执行程序。每次只要你按下 **Step** 按钮，程序会自动转入下一个程序模块。如果你执行**时间延迟或等待**模块，它还可以使程序向下一个模块转换的时间延长。



你还可以使用**下载**操作代替在线操作。在线操作中，程序是由你的电脑执行的。在此模式下，电脑将控制指令，例如“启动电机”传送到接口板。为此，只要程序运行，控制板必须与电脑相连。而在下载操作中，程序是由控制板自己执行的。电脑将程序储存在控制板中。一旦完成，电脑与控制板之间的连接就可以断开了。现在控制板可以独立于电脑执行控制程序。下载操作十分重要，例如在为移动机器人编程时，电脑与机器人之间的连接就十分累赘。尽管如此，控制程序应该首先在线模式下测试，因为那样更容易发现错误。一旦完全测试完毕，程序就可以下载到控制板。使用 **TX** 控制板下载时，**USB** 数据线可以被蓝牙代替，使用 **TXT** 控制板时，还可以使用 **WiFi** 的方式下载。如此一来，模型就可以甚至可以在在线操作下也可以活动自如了（请查看相应控制板的使用手册）。

但在线操作与下载操作相比有很多优点。与控制板相比，电脑有更多的工作内存，因此可以计算更加快速。这对于大程序，是个很大的优点。另外，在线操作中，**ROBOTICS TXT** 控制板，**ROBO TX** 控制板和 **ROBO** 接口板可以同时被一个程序控制。

两种操作模式概览：

模式	优点	缺点
在线	<ul style="list-style-type: none"> ◆程序的执行可在屏幕上显示出来 ◆甚至大程序的执行都很快 ◆多个控制板可以并行控制 ◆支持早先的智能接口板 ◆可以使用面板 ◆程序可以暂停和继续 	<ul style="list-style-type: none"> ◆电脑与控制板必须保持连接
下载	<ul style="list-style-type: none"> ◆电脑和控制板可以在下载后分开 	<ul style="list-style-type: none"> ◆不支持早先的智能接口板 ◆程序的执行无法在电脑屏幕上显示出来

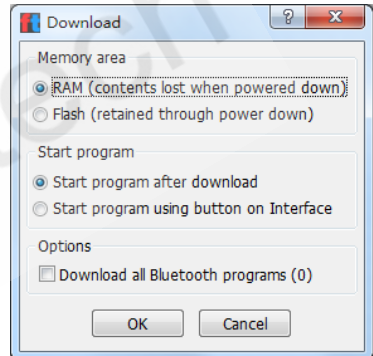
使用下载模式

你可以将车库门控制程序通过

Download(下载)按钮传输到控制板上。

首先，会出现左边的对话框。控制板有好几个程序储存区域，包括一个**随机存取存储器(RAM: Random Access Memory)**和一个**闪存(Flash memory)**区域。

一旦断开控制板与电源的连接或将电池组断电 RAM 中的程序就会丢失。然而对于保存在闪存中的程序，即使断电，也仍然会在控制板中保存好多年。当然，你也可以随时修改闪存中的程序。然而程序下载到 RAM 速度快得多，因此主要在程序测试阶段用。



你可以存储多个程序到闪存，例如一个移动机器人的多个不同动作。你可以使用 ROBOTICS TXT 控制板或 ROBO TX 控制板的显示屏或选择按钮来选择，启动和停止程序。如果选择**下载完成后启动程序(Start program after download)**，程序下载完成后立即运行。

对于移动机器人，“**由控制板上的按钮启动程序(Start program using button on interface)**”选项更有用。因为，如果你没有使用蓝牙或 WiFi，使用 USB 下载程序，在程序启动机器人活动之前，还必须先将 USB 数据线拔除。这种情况下，使用由控制板上的按钮启动程序。

自动运行(Autostart)功能在控制板供电启动后立即开始运行程序，这种情况下，可以使用周期开关电源给控制板供电，使程序在每天的相同时间运行。这样就不需要一直保持控制板供电，或者是手动启动程序。

注意：

你还可以在 **ROBOTICS TXT** 控制板的操作手册中找到更详尽的功能介绍。

Cedutech®

3.8 技巧和诀窍

改变连接线

如果你移动了某一模块，ROBO Pro 会试图以一种合理的方式调整连接线。如果你对某线不满意，你可以方便地通过鼠标左键点击这条线，并按住鼠标键不放来移动这条线。根据鼠标点在这条线上的位置，线的某一角或某一边缘处便会被移动。以下是不同鼠标的用法：



如果鼠标处于一根垂直线上，则可以通过按住左键来拖动整条垂直线。



如果鼠标处于一根水平线上，则可以通过按住左键来拖动整条水平线。



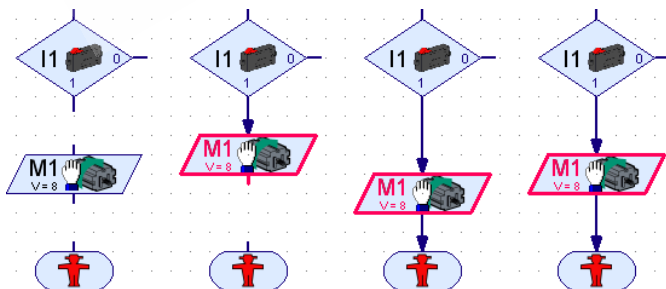
如果鼠标处于一根斜线上，则当在线上左击时，会在线上插入一个新的点，然后你可以通过按住左键来拖动这条线来确定这个新点的位置。



如果鼠标处于线的端点附近或连接线的夹角处，你可以通过按住左键来移动这一点。只能将此连接线的端点移到另一个合适的程序模块的接线端。这样，两个端点就连上了。否则，端点不能移动点。

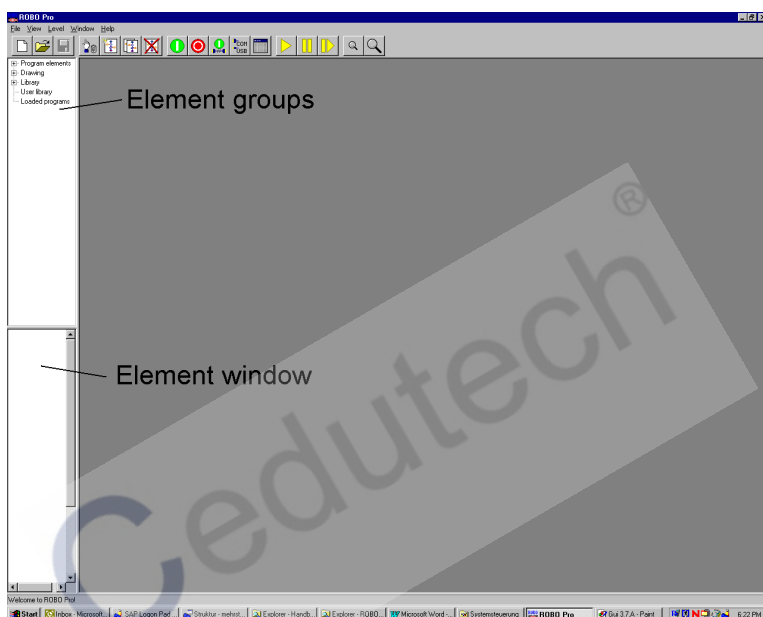
另一种连接的方法

还可以通过移动程序模块来建立连接线。如果移动一个程序模块，使得它的入口位于另一个模块出口下方一到两个格子，就可以建立两个模块间的连线。同样，也适合于将出口移动到入口之上。然后，你就可以将程序模块移动到最终位置。



4 级别 2: 用子程序控制

一旦成功地编写和测试了你的第一个控制程序，你已经为使用 ROBO Pro 级别 2 做好了准备。现在，在“级别(Level)”菜单中选择“**级别 2: 子程序(Level2:subprograms)**”。你会注意到有很多不同。模块窗口消失了，左栏一分为二。



不要担心，模块窗口还在，只不过现在是空的。第二级有更多的程序模块，如果都在一个窗口里显示，可能会顾此失彼的。所以，第二级的所有模块都编成模块组。就像你的硬盘上由文件组成文件夹一样，模块和模块组的关系也是一样。如果你选择左上栏的一个组，属于这个组的所有模块会在下面的窗口出现。你可以在“**编程模块/基本模块(Program elements/basic elements)**”组里找到级别 1 的模块。你可以使用滚动条显示全部的模块。

现在让我们开始真正的主题：子程序！当然我们设计的流程图不会大到一次用到全部模块的规模。但是在大的项目中需要很多灵活的流程图的情况下，这也是有可能的。一旦你的工作表里都是组件的话，到处是连接线，你不得不经常使用滚动条来翻页。那么这样的情况存在吗？混乱！怎么办？有什么办法使这些混乱重归有序？方法就是用**子程序**。

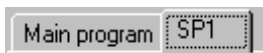
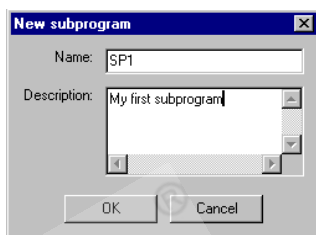
4.1 你的第一个子程序：



子程序和你已经熟悉的程序非常相似。为了理解这点，你先建立一个空的新程序，并在它里面新建一个子程序。方法是，单击程序“**新建(New)**”和工具栏上面的“**新建子程序(SP New)**”按钮。会出现一个窗口，在窗口内你可以输入子程序的名称和描述。

子程序的名称最好不要太长（8 到 10 个字母或数字，最好不要为汉字），不然子程序符号会很大。当然你可以以后来修改的。

一旦你单击 **OK**，关闭了**新建子程序(New subprogram)**窗口，在子程序状态栏上面会显示新的子程序。



任何时候，你都可以点击子程序栏上的程序名在主程序和子程序之间切换。由于两个程序都还是空的，你现在还看不出两者的区别。

现在我们要将上一章（[3.6 “其它程序模块”一节](#)）的车库门控制系统分割成子程序。程序由四个功能模块组成：

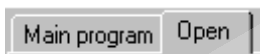
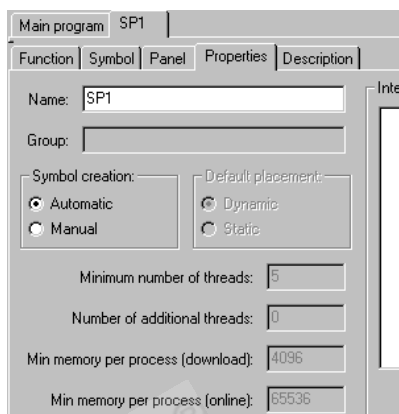
- 等待，直到按下 I1 按钮
- 开门
- 等待十秒钟
- 关门

现在我们要把开门和关门分成两个子程序。任何子程序都可以在主程序中用一个专门的符号调用。“等待传感器 I1 动作”和“延迟 10 秒钟”，这两段仍保持在主程序中，因为它们都只是由一个单个的模块构成的。你已经建立了带名为 **Subprogram 1** 的子程序的新程序，然而“**Open (开门)**”和“**Shut (关门)**”作为这两个子程序的名字应该更好一些。如果你还没有选择子程序 1，你可以通过选择子程序栏上的子程序 1 来重新命名已经建立的子程序。



通过点击**属性(Properties)**，可以从功能栏切换到子程序的属性窗口。在这里你可以将子程序名 **SP 1** 改成 **Open**。其他大多数区域只能在更高级甚至是专家级中才能改变。对于**生成符号(Symbol creation)**这一栏下文将会作出解释。

虽然 “My first subprogram” 是一个正确的描述，但是如果你点击功能栏上的 “**描述(Description)**”，你还是可以改变先前所输入的描述。

现在，点击功能栏上的**功能**，这样你就可以对子程序的功能进行编程。你就会又看到了程序窗口，其中有你上一章创建第一个 ROBO Pro 程序时所插入的程序模块。并确认你已经选择了子程序栏中的子程序 **Open**。

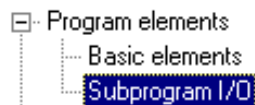


准备好写你的第一个子程序了吗？我们开始吧！主程序总是由一个“开始”模块开头，子程序由一个相似的模块“子程序入口(Subprogram Entry)”开头。模块有这样一个名字是因为控制流程是从主程序经由这个模块进入子程序的。这里你不能用“开始”模块，因为显然没有开始新的流程。

	开始模块	开始一个新的独立的流程
	子程序入口	程序控制由主程序交到子程序

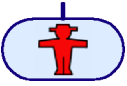



你可以在**子程序入口/出口(Subprogram I/O)**下的模块组窗口找到子程序入口。现在将 **Open** 子程序的子程序入口模块放在程序窗口的顶部。也可以给子程序入口模块取一个区别于 **Entry** 的名字，但这个只是以后你在同一个子程序中用多个入口时才有必要。



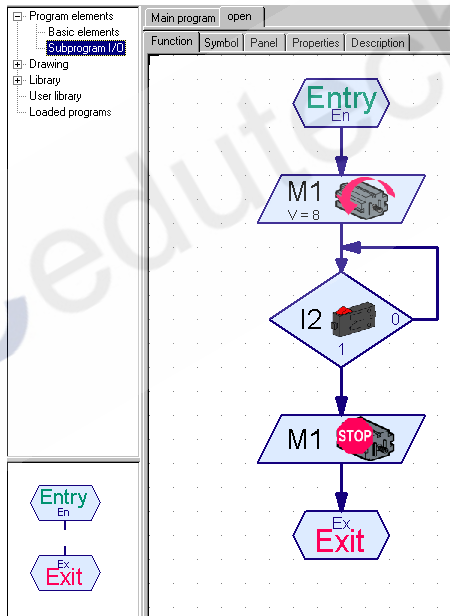
现在，子程序作为主程序的一部分运行，并负责开门。将启动电机 M1 并左转（逆时针），一直等到压下输入端 I2 的限位开关后，再将电机关闭。

你可以用“子程序出口(Subprogram Exit)”来关闭程序。“子程序出口”和“流程停止”模块之间的区别和“子程序入口”和“流程结束”之间的区别是一致的。

	停止模块	停止一个独立流程的执行
	子程序出口	程序控制从子程序交回到主程序



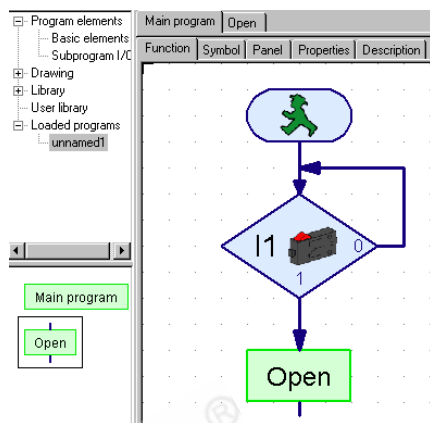
完整的子程序应该如下所示：



确信你是在 **Open** 栏下输入的子程序，而不是在**主程序(Main program)**栏下。现在从 **Open** 栏切回到**主程序(Main program)**栏。就看到了主程序窗口，和先前一样还是空的。通常，可以先插入一个“开始”模块（不是子程序入口）到主程序。并查询用来打开车库门的开关 **I1**，和先前主程序中所作的一样。

现在你可以将你的新子程序插入主程序（也可以是其它子程序），方法和普通程序模块一样。你可以在 **Loaded programs** 栏下的模块组窗口中找到它，还有你的程序名。如果你还没有保存过你的文件，则其文件名为 **unnamed1**。如果你还打开了其它程序文件，你也可以在选择其它程序文件的子程序。用这种方法，可以很方便地使用其它文件的子程序。

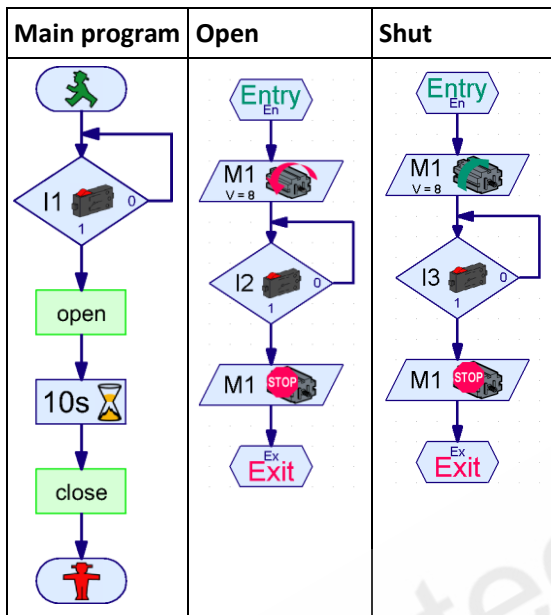
在模块组 **Loaded programs / unnamed1** 一栏中，可以找到两个绿色的子程序符号。首先，名字 **Main program** 代表的是主程序的符号。它很少用作子程序，但还是有可能的，例如你正在控制整个的工厂，且你先前已经为各个单个的机器的控制系统开发了主程序。第二个符号的名字代表了你的新的子程序，**Open** 这个名字是在属性窗口中输入的。现在，将子程序符号和普通程序模块一样插入主程序。



如果你喜欢，可以立即用一个停止模块来将主程序终止，试一下吧。现在可以通过按一下按钮 **I1** 来将门打开，但是我们还没有对关门部分进行编程。为此，我们来写另一个子程序。按一下工具栏上的 **SP New** 按钮，并在 **New subprogram** 窗口中输入子程序名 **Shut**。描述输入不是必须的，只是之后别忘了这个子程序的用处。

现在，可以在程序窗口中输入车库门关门子程序 **Shut**。又一次从“子程序入口”开始。首先电机应该向右转（顺时针）。限位开关 **I3** 一旦闭合，电机 **M1** 应该停止。子程序还是用“子程序出口”结束。

现在通过子程序栏切换回主程序。如果你先前已经用停止模块结束了主程序，必须再将停止模块删除。车库门打开之后，应该再关闭并保持 10 秒钟。在一个 10 秒钟的延时模块之后，你可以从 **Loaded programs / unnamed1** 程序组中选择 **Shut** 子程序插入主程序。主程序和两个子程序具体如下所示：

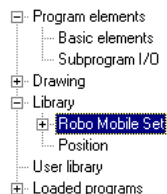


主程序由开始模块开始执行，然后等待直到 I1 传感器被按下。顺便提一下，这里也可以用“等待输入(Wait for input)”模块([见 8.1.9 “等待输入”一节](#))。开关 I1 按下之后，主程序调用子程序 **Open**。这样，程序控制转到了 **Open** 子程序的子程序入口。子程序将车库门打开，然后到达子程序出口。这时，程序控制又转回了主程序。子程序结束后，主程序又等待了 10 秒钟。然后程序执行到车库关门子程序 **Shut**。程序控制从子程序回到主程序之后，主程序来到停止模块，程序结束。



4.2 子程序库:

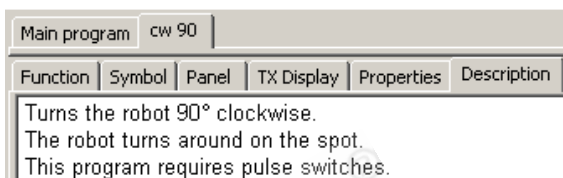
把子程序从一个文件复制到另一个文件十分方便: 同时加载两个文件, 用**已经加载的程序模块组(Loaded programs)**, 将子程序从一个文件插入到另一个文件。对于经常使用的子程序, 使用**库(Library)**的话, 操作会更简单。ROBO Pro 包括了一个预置的子程序库, 你可以方便地重复使用。使用同样的方法, 你可以建立自己的库, 保存你经常使用的子程序。



4.2.1 使用库：

库初始分为两个主要的组。在**组合包(construction kits)**组里，可以找到用于特定的组合包里的模型子程序。在**扩展(Advanced)**组里，可以找到能用于所有模型子程序。但是在扩展组里的大多数子程序需要级别 3 的功能，在下面一章会解释。

如果你用鼠标指向其中的一个子程序符号，会显示一个简短的描述信息。如果你将一个子程序插入程序，可以在子程序工具条上选择子程序，再单击功能工具栏的**描述(Description)**，就可以显示其详细描述。



注意：如果你从库里插入一个子程序，在一些情况下，比如某个子程序所用到的子程序会同时插入程序。你可以选择编辑菜单上的撤消功能，删除这些子程序。

4.2.2 使用你自己的库:

使用了一段时间 ROBO Pro 软件后,你一定会经常使用某些子程序。为了避免每次使用子程序都要查找打开相关文件,你可以建立你自己的库,就像预置的库一样。你的库由存放在同一个文件夹下的 ROBO Pro 文件组成,该文件夹下的每个文件都归类到自己的组中去。

你可以在**文件(File)**菜单的**用户自定义库文件路径(Own library directory)**选项里,定义在哪个文件夹里存放你自己的库。默认的路径为 C:\Programs\ROBOPro\Own Library。如果你在计算机上有自己的个人目录,你可以把你自定义库的文件夹定义在那里。

小贴士:开始时,你可以把自己的库文件定义在存放你的 ROBO Pro 程序的那个目录。用这种方法,你可以在你的目录下,快速的找到所有文件中的所有子程序。

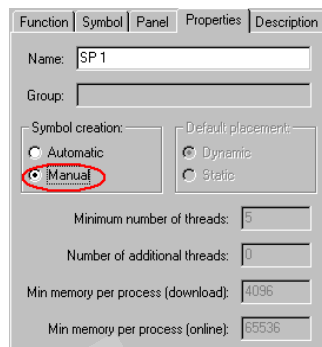
组建你自己的库:

ROBO Pro 没有特别的改变库的功能,不过操作起来也很方便。如果你需要从一个库中添加或者删除子程序,你必须先调用相关的文件。可以在你**自己的库文件目录**下找到这个文件。现在,你可以打开第二个文件,在“**已加载的程序(Loaded programs)**”组中选择一个子程序,然后拖动到库的主程序中。在库里,主程序不是一个真正的程序,只是库里所有子程序的集合而已。在库里,主程序本身并不显示在模块窗口。当然,你也可以在那里修改或者删除子程序。

如果你修改了一个库文件并保存,还必须在“**文件(File)**”菜单里选择“**更新我自己的库(Update own library)**”。这样可以更新组窗口的文件列表。

4.3 编辑子程序符号

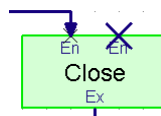
就如你在前一个章节所看见的，ROBO Pro 会自动为你的子程序生成子程序符号。但是你也可以自己定义符号，用来更好表达此子程序的功能。要这么做，你需要在子程序的属性窗口，从自动符号模式切换到手动符号模式。接下来，你可以从“属性(Properties)”栏切换到“符号(Symbol)”栏，来编辑子程序的符号。这个模块组窗口中的“绘制图标(Draw)”模块组中可以找到绘图功能。



在画图/形状(Draw/Shapes)这个模块组里，你可以找到常用的图形，如矩形，椭圆，圆形，多边形等等。在画图/文字(Draw/Text)下面，你可以找到各种字体和大小的文字对象。在画图的其它组里，你可以找到更改颜色和其他属性等功能。如何使用绘图里的功能，在[第十章绘图功能](#)里会详细描述。绘图功能也可以在编程窗口使用。

你可以移动子程序的接线端，但是不可以添加或者删除。在子程序符号的环境下，一般每个子程序的入口和出口只有一个接线端。接线端是自动生成的，即使你切换到了手动符号模式。

一旦你离开符号编辑窗口，所有主程序和其他的子程序下的该子程序都会相应地自动修改。请注意以下情况，如果你已经移动了子程序的连接端，且连接已经建立的话，这会对子程序的调用发生一点影响。在某些环境下，会出现不正确连接，而是在连接线的端点和符号端点都出现一个叉（见图）。作为惯例，一般简单地在连接线上单击鼠标左键，线将自动重画。在子程序有许多连接的情况下，就必须以后编辑这些线。



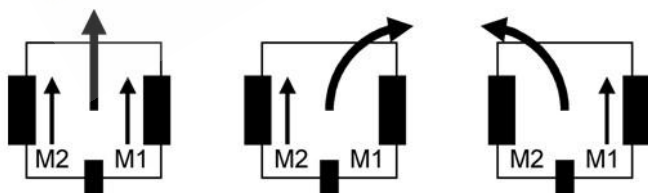
4.4 TANGO 控制

目前为止，你只熟悉简单的程序，也许您正在热切地期待新的程序单元和可能性。但是，在我们处理下一章变量和更困难的事情之前，让我们先看下在级别 2 环境下能做哪些事情。例如，如何将探戈舞程序加载到您的移动机器人上？让“Nerdse”机器人在你们中间，随着音乐跳起绅士步伐的探戈舞，该步骤包含 3 个阶段，共 8 个步骤，顺序如下：

- 左脚向前一步（1/4 节拍）；
- 右脚向前一步（1/4 节拍）；
- 连续 4/8 节拍的“摇摆步”，摇摆过程中，小步移动，控制重心，首先左脚移动 1/8 节拍，然后右脚移动 1/8 节拍，接着左脚再移动 1/8 节拍，完成摇摆之后休息 1/8 节拍；
- 接着 3 个步骤如下：首先，右脚回一小步，靠近左脚，然后左脚让一步，随后右脚再次靠近左脚，以上三步各占一个 1/8 节拍，最后以一个 1/8 节拍停顿结束。

女士步伐顺序是对称的，也就是左右交换，前后交换。重复上述步伐直到音乐结束，或者撞到房间的边缘，或者你觉得枯燥乏味了，结束时，你应该寻求舞蹈大师的意见。

现在轮到机器人出场，或许你有一个慧鱼移动机器人组合包，组合包包含两个驱动轮，分别由电机驱动，运动方式与机动车相似，即：电机转向相同，机器人直线运动，一个电机转动，一个电机停止，机器人转弯，电机转向相反，机器人原地掉头。



现在，让我们尝试编辑探戈舞步伐，1/4 节拍对应电机旋转一圈：

第一阶段：

- 左轮向前 1 圈（通常是电动机 M2 左）。
- 右轮向前 1 圈（通常是电动机 M1 左）。

第二阶段 摇摆步事实上，机器人无法保持轮子转动而身体不动，同时侧向移动对于机器人来讲，难度比较大。因此在第二阶段中用轻微的转动模拟摇摆，在第三阶段中用小幅度侧向前进模拟侧向让步。第二阶段顺序如下：

- 左轮向后 1/2 圈（通常是电动机 M2 左）。
- 右轮向前 1/2 圈（通常是电动机 M1 左）。
- 左轮向后 1/2 圈。

第三阶段，右上左退阶段顺序如下：

- 右轮向后 1/2 圈。
- 直线向前 1/2 圈。
- 右轮向后 1/2 圈，左轮向前 1/2 圈。

因此，我们首先设定机器人小幅右转，然后沿左边前行，模拟左侧步伐，之后机器人恢复直线。现在，让我们试着在 ROBO Pro 软件中，执行以上步骤。执行的形式取决于您是否正在使用编码电机或脉冲开关模式控。这两种情况分别在 [4.4.1 带脉冲开关的电机控制](#)和 [4.4.2 带编码器电机的电机控制](#)进行叙述。

4.4.1 带脉冲开关的电机控制

最好将每步编写成子程序，第一步“左轮转 1 圈”的子程序如下图所示左侧图所示。通常情况下，左轮驱动电机连接到 M2 输出接口，相应的脉冲开关连接 I2 输入接口，这时电机逆时针旋转方向为前行方向。

第一步，切换电动机 M2 逆时针（全速），然后，等待 I2 输入接口输入 8 个脉冲，8 个半脉冲计数意味着开关 0→1 和 1→0 阶跃都记录。可以在属性窗口选择脉冲计数器的类型，许多模型中 8 个半脉冲对应车轮旋转 1 圈。但是，这也可以改变，选定不同的传动比和不同的脉冲开关布置位置，例如每圈 16 半脉冲。

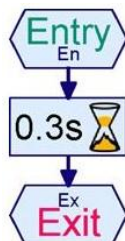
输入 8 个脉冲后，开关断开，M2 电机停止。你可以调用这个子程序，完成左轮前转 1 圈。”

为完善主程序，你还需要以下子程序：

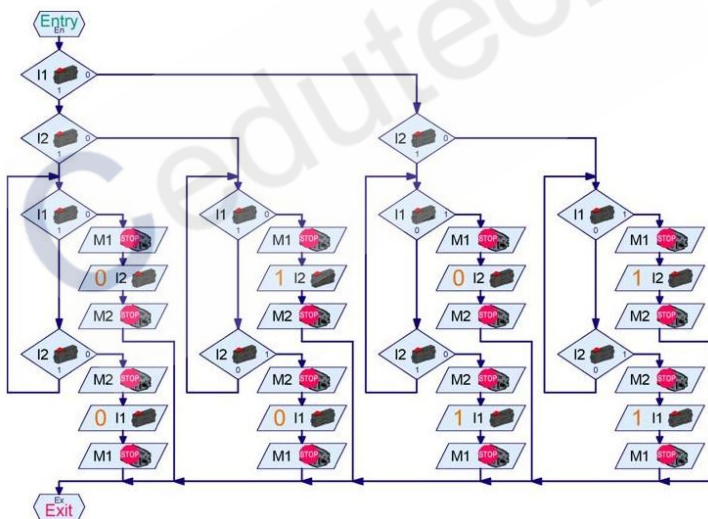
- 右前转 1/4 节拍（与左转相同，用 M1 和 I1 代替 M2 和 I2）
- 左后转 1/8 节拍（如同左前转 1/4 节拍，但是用 4 个半脉冲代替 8 个半脉冲，同时反向，因此，电机顺时针转动）
- 右前转 1/8 节拍（如同右前转 1/4 节拍，用 4 个半脉冲代替 8 个半脉冲）。
- 右后转 1/8 节拍（如同右前转 1/8 节拍，但是反向，因此电机顺时针旋转）。

当然，你也可以不通过计数器暂停 1/8 节拍脉冲，相反，我们使用延迟时间实现以上目的。ROBO 移动机器人标准模型中，4 个半脉冲对应 0.3 秒左右，因为电机和模型的不同会有差异。上图右侧图为延时子程序，子程序除了包含子程序入口与出口之外，只包含一个程序指令，程序中你需要用到延时子程序 2 次。如果你用子程序来控制延时，就可以轻易地改变延时时间。

现在，也许你会认为我们应对每步添加延时程序，取代计数功能，这样就会使每步的时间与暂停时间匹配。延时程序的缺点是右电机和左电机转速无法达到高度一致，致使机器人不能重现舞步。如果使用脉冲开关，相反，即使是蓄电池电量不足或者一个轮子比另外一个轮子转动困难，你也可以确保这两个轮子转速完全相同。



为爱问问题的各位，下面将会对“SyncStep”子程序做简短的介绍。对子程序充分了解的人们可以跳过下面的内容。不明白子程序的工作原理也不要紧，只要你明白如何使用子程序就行。



通过输入等待指令等待，直到第二个输入变为 0，然后关闭第二个电机。因为你不知道这两个电机哪个转动的更快，以及脉冲开关的变化，两个输入的等待循环是十分必要的。其他 3 种情况完全如此。他们从其他初始状态开始，等待的最终状态分别与各自的初始状态相反。例如，左边的第二个分支中，在开始时 $I1 = 1$ ， $I2 = 0$ ，这个初始状态你可以很容易地通过前两个分支指令检测到。从而，第二个分支等待 $I1 = 0$ ， $I2 = 1$ 。如果你想自己写程序，你必须非常注意每个分支中脉冲开关的初始值，以及每个分支中需要等待的相应数值。

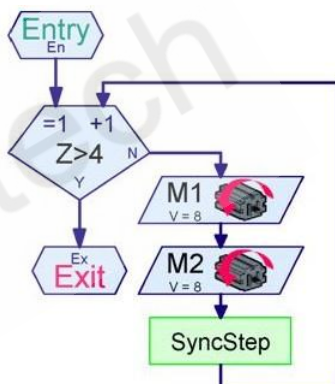
如果您已经浏览过以下章节中的变量问题，可以尝试写一个相同功能的变量子程序。这是比较容易的，可以在开始命令中用两个变量保存两个脉冲开关的值，而只需要一个程序分支，这样就可以比较变量的变化，确定分支情况。

所以，现在回到探戈步伐程序：前进 4 个半步的“1/8 拍直线前进”子程序中插入 SyncStep 子程序的目的。如果启动电动机 M1 和 M2，执行 SyncStep 的子程序，电机前进半步后再次停止。因此，你必须设定子程序执行 4 次，最佳运行方式就是采用循环变量。

如果你十分细心，你现在可能担心电机在 SyncStep 子程序执行完停转后，是否能够立即启动。两个电机中相对转速慢的一个在开关过程中有一个轻微制动，这对于两个电机之间转速的相互匹配是十分必需的，然而这对电机本身是有害的。事实上，控制板也是通过不断地开关电机调节电机的转速，这被称为 PWM（脉宽调节）控制。相反，对于高速电机，关闭开启发生如此之快，以至于电机反应不过来。所以，在 SyncStep 子程序中，你也可以放弃关闭第二个电机，而是在循环结束后尽快关闭两个电机。由于编程方法多样，往往可以采用不同的方式实现这一目标。

测试一下机器人是否按照 SyncStep 子程序实现直线运行，是否比让电机简单地运行固定脉冲数的效果好。

最后一个子程序，我们需要机器人完成原地向右转 4 个半长的步。有趣的是，您可以使用“1/8 直线”子程序中完全相同的“SynchStep”子



程序。“SynchStep”子程序仅让电机停转，而停转命令不依赖于旋转方向。启动循环中的电机 **M1**，简单用右转代替左转。脉冲开关与转向无关，电机正反转表现到脉冲开关上都一样，**0→1**，**1→0**，控制电机转向。因此，创建“转 1/8 拍”子程序只需要复制“1/8 拍直线前进”子程序并改变电机旋转方向。

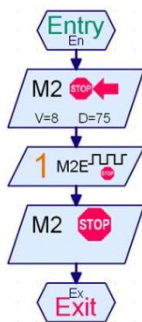
Cedutech®

4.4.2 带编码电机的电机控制

你最好开始使用子程序实现各个步骤，下图为“左轮转 1 圈”的子程序。通常情况下，左轮连接到 **M2** 输出接口，相应的脉冲开关连接到 **C2** 输入接口，电机逆时针旋转。

第一步，开启电机 **M2**（全速顺时针旋转），然后等待 **C2** 输入 75 个脉冲。75 个完整的脉冲意味着，等待 1 到 0 的变化 75 次和 0 到 1 的变化 75 次。编码电机可以设定脉冲数控制电机，在属性窗口中选择运行距离，在距离选项处设定为 75 个完成脉冲。

M1E 命令无需等待电机运转达到其目标，电机运行时，程序可以做其他事情，然而，这里我们就是要让电机运转到指定目标。任务完成后，每个电机的输出都有自己的“目标”输入，如电机 **M2** 的输入为 **M2E**。



编码电机连接到 **TX** 控制板时，电机需要同时连接到 1 个输出接口（**M1-M4**）和 1 个计数器输入接口（**C1-C4**）。编码电机默认相同编号的计数器输入和电机的输出，这就是为什么不能在高级电机控制属性窗口中调整计数器的编号。

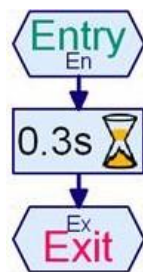
电机完成任务后，删除运行距离，电机不再执行向左或向右等一般命令。要做到这一点，需要使用高级电机控制，设定电机停止。然而，这仅仅适用于用通用电机指令控制电机，电机响应电机高级控制命令时，无需停止运行指令。

为完善步骤，你需要下面的子程序：

- 右前转 1/4 拍（如同左前转 1/4 拍，用 **M1** 和 **M1E** 代替 **M2** 和 **M2E**）
- 左后转 1/8 拍（如同左前转 1/4 拍，37 个脉冲代替 75 个半脉冲，电机顺时针旋转）
- 右前转 1/8 拍（如同右前转 1/4 拍，用 37 个半脉冲代替 75 个半脉冲）
- 右后转 1/8 拍（如同右前转 1/8 拍，但是电机顺时针旋转）

当然，因为暂停时停止转动，不能使用脉冲计数器计算 1/8 拍脉冲，相反，我们使用了延时功能。**ROBO TX** 综合训练组合包的标准模型中，37 个脉冲对应约 0.3 秒，会因为模型传动比和电机差异而有所不同。所以，你需要编写一个 1/8 拍暂停的子程序。除了子程序的输入和输出，子程序只包含一个程序指令，但你需要暂停两次。如果创建完子程序，那么你很容易改变暂停时间。

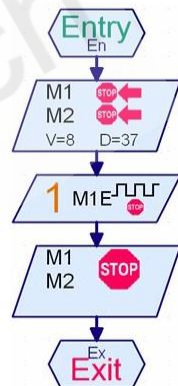
现在，你可以说，我们也应该使用延迟时间代替电机的高级控制。这将避免暂停时间和每步运行时间的匹配问题。但缺点是，无法确保左右电机以完全相同的速度运转，从而机器人无法重现舞蹈步伐。另一方面，采用电机高级控制，确保两个轮子总是前进完全相同的距离，即使电池电量不足和齿轮差异等问题。



现在我们只需要“直线前进 1/8 拍”和“原地旋转 1/8 拍”的子程序。电机高级控制还提供了同步和同步距离操作，实现同时控制两个电机的可能性。同步控制确保编码电机以完全相同的速度转动，结果你的机器人几乎笔直向前移动。然而，由于车轮有一定的滑动，编码电机也无法实现精准定位。在我们的例子中，M1 和 M2 以相同的速度运行 75 个脉冲的距离。要做到这一点，可以使用远程同步命令。为了达到电机同步耦合，直到两个电机任务完成后，才发出任务完成信号。因此，对于两个任务完成信号中的一个要有足够的等待，最后不要忘记停止两个电机！

多次尝试，观察使用同步距离操作后，对比单独对每个电机进行距离控制操作，机器人是否真的移动更加准确。

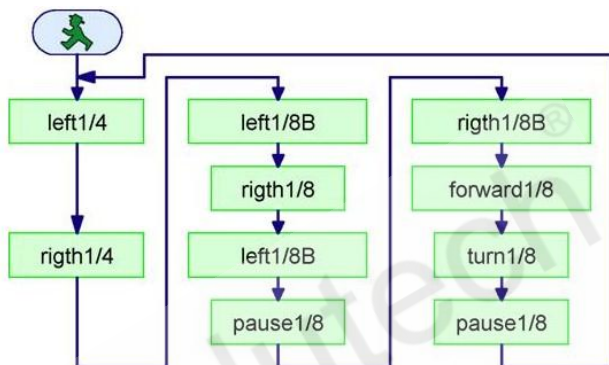
最后，我们需要机器人停止后右转 37 个脉冲的子程序。在这里，再次使用电机高级控制与同步距离命令，为了完成“1/8 拍转向”子程序使两个电机反向转动，想想 M1 和 M2 应如何转动实现模型原地右转。



4.4.3 TANGO 主程序

现在，在编写好所有子程序后，你可以开始编写主程序了。这时问题变得不那么困难，下图为主程序直观图。

下图所示为主程序运行探戈舞的循环步骤，尝试使用循环计数功能，在 1 次循环中进行 5 次探戈舞步。为了这个目的，运用复制和插入功能，将主程序内容复制到 1 个新的子程序中，并添加 1 个输入和输出的子程序，然后你可以在 1 次循环中运行 5 次这个子程序。



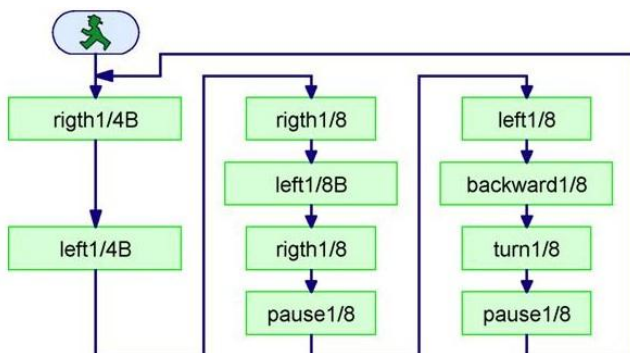
您可以在 ROBOPro 安装列表下找到探戈舞程序：

Sample programs\Manual\Tango Encoder Motor\TangoSolo.rpp

Sample programs\Manual\Tango Pulse Switch\TangoSolo.rpp

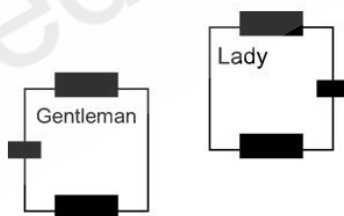
然而，如果你有一个合适的机器人，可以用自己编写的程序或者已有的程序测试下。

现在，也许你会想：这是相当不错的，但实际上探戈舞需要两个人完成，事实上，男女步伐是对称的，编写一个女士探戈舞步的程序并不难。首先打开男士舞步程序，另存为一个新的名称为 **TangoSoloLady.rpp** 的文件。然后改写子程序，例如，**向前左转 1/4 拍**改为**向后右转 1/4 拍**，你必须将 **M2** 改为 **M1**，同时改变旋转方向。点击“属性”选项卡，重新命名子程序，此后主程序中，子程序的名称会自动变更。



“转动 1/8 拍”子程序不需要改变，你知道吗？改变左右转向（调整子程序中 M1 和 M2），在子程序中调整电机转向以改变机器人向前和向后的步伐。改写后的子程序与原子程序进行对比，看看是否发生变化。

如果你有两个移动机器人，现在一个加载 **TangoSolo.rpp** 程序，一个加载 **TangoSoloLady.rpp** 程序。如果你只有一个机器人，你需要另外寻找一个机器人配合。程序下载时，你应该说明程序是在接口板上开关开启后才开始运行。现在，将两个机器人如下图对置放，为了简单演示，需要通过开关控制（TXT 控制板或 TX 控制板），同时启动两个机器人。



如果同时启动两个机器人后，两个机器人各自都完美运行，但是两个机器人舞步无法严格对称，这是由于电机和蓄电池不是完全相同，电机不能迅速反应做出精准动作，之后两个机器人迟早会出现明显区别。如何实现两个机器人，在较长的时间内保持一致，你将在下一章学习到。

4.5 TANGO 控制 2

通过蓝牙或射频数据链路通信

为了保持节奏，这两个机器人需要协调他们的探戈舞步。TX 控制板具有一个集成的无线蓝牙模块，ROBO 接口板具有 ROBO 射频数据链路。ROBO 射频数据链路包括两个无线模块，其中一个无线模块内置卡直接嵌入 ROBO 接口板，PC 无线模块通过 USB 连接到 PC 机上。到现在为止，你可能只用无线电链路管理您的移动机器人，因此无需电缆在线连接。但蓝牙和射频数据链路可以做更多的事情：两个机器人可以交换消息，彼此沟通。

在级别 2 编程环境的编程模块选项中，有**发送和接收(Send,Receive)**两种命令符，如图所示：

左图中的框图是发射器，右边的框图是接收器。



表示传输的元件通过缩写为 FRN 1 的 1 号无线呼叫信号将 Hello 消息发送给 TX 控制板或 ROBO 接口板（下面只是 TX 控制板）。无线编码类似电话号码，表示控制板发送消息的手段。你将在 [4.5.1ROBO 接口板无线设置](#)和 [4.5.2TXT 和 TX 控制板的蓝牙设置](#)获得更多信息。

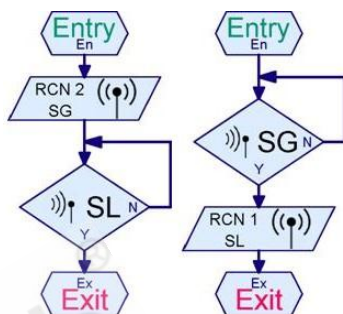
在图片右侧的接收器命令就像一个分支程序：如果收到 Hello 消息，该命令到输出 Y 分支，否则到输出 N 分支。

我们假设无线通信代码为 2 的控制板上的程序执行发射模块，然后发射模块向无线通信代码为 1 的控制板发出 Hello 消息，发射模块中已经设定 1 号控制板作为目标控制板。无线通信代码为 1 的控制板表明 Hello 消息已被接收。如果下一次接收元件向控制板发出 Hello 消息是否接收到的查询指令，答复先为“YES”，然后为“NO”，直到另一个 Hello 消息被接收。如果这个控制板接收模块再次询问是否接收到 Hello 消息，回答先为“YES”，然后为“NO”，直到接收到新的 Hello 消息。接收模块不能区分是哪个控制板发送的信息，为了能够区分，你必须发送不同的信息。

发送的信息是一个任意字段，例如“**Hello**”。然而，由于技术原因，只有信息的前三个字母或数字能够被准确识别，你可以表明更多的特征，但是“**Hello**”、“**Help**”和“**helicopter**”将代表同一个信息，因为它们都以“**Hel**”开头。同样不区分大，小写和特殊字符（空格符，！，？，%之类），如 **XY!** 和 **XY?** 代表相同的信息，但能够区分数字字符，**XY1** 和 **XY2** 却代表不同的信息。

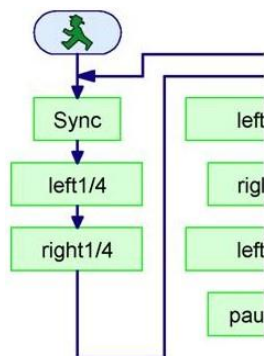
现在，同步两个拥有发射器和接收器的探戈机器人并不困难。循环开始，一个机器人发送“我们可以跳舞吗”的信息，另一个机器人接收到之后发送“让我们跳舞吧”的信息，然后开始舞步。由于消息不需要这么长，我们称“**SG**”为男生步伐，称“**SL**”为女士步伐。下图为两个同步子程序框图，左侧子程序代表男士机器人，其由 2 号无线电控制，男士机器人先发送“**SG**”信息给女士机器人，并等待女士机器人发出“**SL**”信息。

右图为发射器属性窗口，在“发送指令(send command)”栏中可以从列表选择一个命令，或输入自己的命令，在“目标接口板(destination interface)”栏，你可以选择命令信息通过特定的无线电发送到一个控制板或所有控制板。



左图为接收器属性窗口，在“接收指令(receive command)”中，输入接收命令信息，然后，您必须进一步选择接收器是否通过特定的无线电接收直接收到的命令信息，还是接收群发的命令信息。最后，你也可以设定接收信息后“Y”、“N”的分支路径。

到目前为止，我们已在发射器和接收器属性窗口中命名“消息”。在级别 3 编程环境中，将使用其中的“command”选项。从数据传输角度来看，这是相同的。信息是否是一个命令，取决于释义，而不是哪种传输。在级别 3 编程



环境中，你将使用很多“messages”，例如，控制电机或变量命令。因此，在 ROBO Pro 软件中，“command”选项通常用于信息。

现在，将前面编写好的同步子程序插入到 TangoSolo.rpp 和 TangoSoloLady.rpp 主程序之中，同步子程序命名为“sync”。你也可以按照同样的方式，编写女士邀请男士跳舞的子程序。如图示主程序中的子程序，你可以随时在循环开始调用。你可以在 ROBO Pro 安装目录下找到相关子程序：

Sample programs\Manual\Tango Encoder Motor\

Sample programs\Manual\Tango Pulse Switch\

TangoSyncGentl.rpp

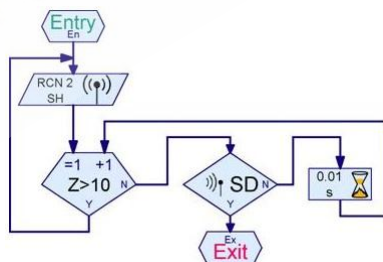
TangoSyncLady.rpp



将两个程序分别下载到两个机器人，并启动程序。通过 ROBO 接口板和数据链路，你会看到只有“女士”开始跳舞后，男士才开始跳舞。这是因为男士舞蹈机器人首先发送一个“SG”消息，女士舞蹈机器人等待该消息。如果男士舞蹈机器人先启动了，邀请信息为空，女士舞蹈机器人将一直等待，相反，如果女士舞蹈机器人先启动了，她已经接收到男士舞蹈机器人发送的“SG”信息。这自然有些不实用，特别是当你忘了哪个机器人是男士，哪个是女士。



TX 控制板不会出现此问题，因为男士机器人直到接收到女士机器人的蓝牙信号，才会发送消息。



很简单：男士机器人必须重复发出邀请，等待一段时间，直到得到女士机器人的回复。任何时候当男士机器人发送邀请信息，他都在等待女士机器人的答复。在右边的流程图中，男士机器人发出“SG”邀请消息，然后等待 10 次 0.01 秒循环，如果在这段时间内没有得到答复，男士机器人再次发出邀请。



现在你或许会问自己为什么男士机器人不能在循环中简单地发送一个“SG”信息，并立即检查是否收到了“SL”信息。这是因为信息从男士机器人传递给女士机器人需要 0.01-0.02 秒，往返时间相同。即使女士机器人的子程序已经运行，控制板收到信息也需要 0.04 秒。在这段时间内，男士机器人可以发送批量的“SG”信息，这些信息都将传输给女士机器人。由于第一次同步没有起作用，女士机器人的接收器中保存有过量的“SG”信息。下一步循环开始时，在男士机器人发送新邀请之前，女士机器人已经接收到“SG”邀请信息，这时女士机器人不在等待男士机器人的邀请。因此，重新发出邀请信息之前，男士机器人必须等待足够长的时间。



自然，信息发送完成后，男士机器人也等待 0.1 秒，检查是否收到回复。设定 10 次 0.01 秒的循环等待，比男士机器人收到女士机器人答复后立刻继续运行程序有优势。现在尝试一下如果你如上述改变 TangoSyncGentl 程序中的同步子程序，程序是否运行良好。无论首先启动哪个机器人，第二个机器人总会立即启动做出反应。



如果你让机器人跳舞，直到电池电量耗完，同步子程序将不能很好运行。事实上，在每一循环开始，机器人都进行同步，但在循环过程中，如果其中一块电池电量过低，他们的舞步将会明显不同步。最好在每一步中插入一个额外的同步子程序。这里很明显有两个程序都在运行，你就不能重复。因此，初始的同步与循环过程中的每步就不应混淆，你需要使用两个不同的子程序“Sync1”和“Sync2”，这两个子程序将使用不同的信息，例如 SH1, SL1 和 SH2, SL2。可以在 ROBO Pro 安装目录下找子程序：



Sample programs\Manual\Tango Encoder Motor

Sample programs\Manual\Tango Pulse Switch

TangoGentl.rpp

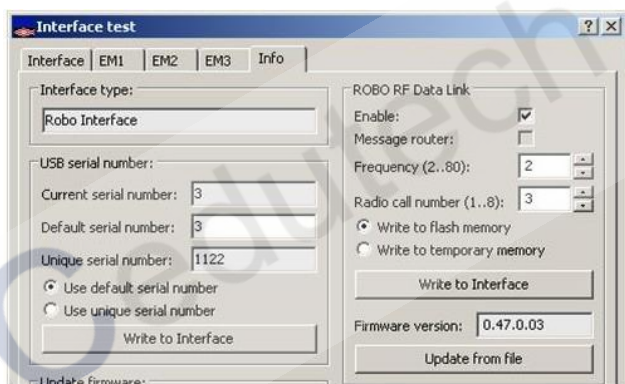
TangoLady.rpp

你会发现有一个 TangoNachrichtenMonitor.rpp 程序，如果你还有一个 TX 控制板或带射频数据链路模块 ROBO 接口板，你可以在在线模式下启动程序，控制两个机器人同时跳探戈舞。屏幕上的程序会显示发送的是哪条信息。该程序采用级别 3 编程单元，现在你还不需要了解该程序是如何工作的。

从级别 4 编程开始，有一组接收器实现此功能。

4.5.1 ROBO 接口板无线设置

你可以在“interface test-info”属性窗口命名每个 ROBO 接口板编号（1-8 之间）和与之相应的信号频率，上述设定可在下图右上方“ROBO RF Data Link”中调整。所有交换信息的 RF 数据链路模块必须为相同频率，输入频率为 2-80 之间的数。如果同区域有几组机器人，你可以改变频率，例如，在一所学校或一场比赛，需要机器人彼此独立传输数据。同组的机器人使用相同的频率，不同组使用不同的频率，如果 RF 射频数据链路不稳定，您也可以改变频率。许多无线电系统，例如无线 PC 网络与 ROBO 射频数据链路，使用相同的频率范围（2.4 千兆赫）。如果 RF 射频数据链路被其他无线电系统干扰，调整频率范围可以解决问题。但是请注意，你必须改变所有的 RF 数据链路模块和 PC 无线模块，必须保证使用相同的频率。



所有包含射频数据链路的 ROBO 接口板，必须有独立的编号为 1 到 8 的无线电，同时要有相同的频率。编号 0 被保留给 PC 无线模块，因此，最多 8 个 RF 无线射频数据模块可以与 PC 无线模块互相沟通。类似于电话号码网络，您可以指定 1-8 之间的任意数字给 8 个接口板。

上图中“Enable”选项总是勾选，然而，如果你根本不使用，您可以停用无线模块接口，在不拆除模块情况下节省电流。

你已经做了一切的调整后，你可以使用“write to interface”按钮写入保存接口板的调整。作为一项规则，你的调整写入“Flash memory”，然后保存，即使关闭接口板。如果你只是想简单地尝试一下，你可以将调整写入“temporary memory”。

你不必关心“firmware version”，这是 RF 射频数据链路内部控制程序的版本。必要的时候，ROBO Pro 会提示您自动更新硬件。

为了实现两个探戈机器人通信，首先用 USB 数据线将 PC 机与接口板，打开接口板测试窗口。选择“interface test”之前，先点击“COM-USB”按钮，选择连接方式。然后在接口测试窗口，调整接口板编号为 1 和无线信号频率为 2，然后用“write to interface”按钮保存状态。关闭“interface test”窗口，连接另外一个接口板，调整接口板编号为 2 和无线信号频率为 2，完成检测。

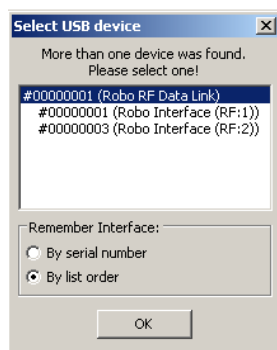
4.5.1.1 无线电通信

两个探戈机器人的情况下，控制板通过射频数据链路模块直接交换信息过程中，红色 PC 无线模块是必要的。红色 PC 作为所有模块的交换中心，因此必须连接到电脑上。电脑必须开机，避免进入休眠和睡眠状态，确保 PC 无线模块实时运行。PC 机的无线模块必须与控制板无线模块的频率相同。PC 无线模块编号为 0，不能改变。

为了调整 PC 无线模块的频率，连接 PC 模块到 USB 接口。然后，你可以按照接口板频率调整方法，在“interface test”窗口调整 PC 无线模块频率。如果 PC 无线模块不能连接到控制板内置无线模块，例如，当你打开“interface test”窗口，频率不正确，您会收到一条错误信息。该错误消息指的是没有发现控制板，即并没有输入和输出可用。然而，尽管出现错误消息，你同样可以在“info”窗口进行调整。

4.5.1.2 通过接口（COM / USB）按钮选择接口

到目前为止，你可能大多只使用一个 ROBO 接口板。当多个 ROBO 接口板或 PC 无线模块与 PC 机连接，控制板在线模式启动，下载程序到控制板，或打开控制板测试窗口时，ROBO Pro 软件将连接哪个控制板？点击“ROM-USB”窗口，首先出现接口板选择窗口。当你选择 USB 连接，ROBO Pro 软件就会通过 USB 线或无线网络找到不止一个控制板，选择窗口如下图所示。在这个例子中，电脑无线模块的射频数据链路连接到 PC 的 USB 接口，电脑无线模块已发现两个 ROBO 接口板，其中有无无线编号为 1 和 2，在这个窗口中你可以选择操作的控制板。作为一项规定，你可以选择两



块控制板其中之一，而不是 ROBO 射频数据链接。



如果您选择上图中的射频数据链接，ROBO Pro 连接到最小的无线电编号，例如 1。然而有个重要的区别：当选择射频数据链接，“interface test”窗口的信息将与 PC 无线模块自动匹配，而不是通过无线电与控制板无线模块连接。因此，你可以通过无线电，改变控制板无线模块和 PC 无线模块的频率，而无需将控制板通过 USB 电缆连接到 PC 机。为了这个目的，首先通过“COM - USB”按钮，选择其中一个控制板，通过“interface test”窗口，改变控制板的频率，并再次关闭“interface test”窗口。如果你现在再次按下 COM / USB 按钮，改变了频率的控制板从列表中消失，这是因为该控制板频率不能被 PC 无线模块识别。选择第二块控制板并改变其频率，如果你再次按下“COM - USB”按钮，就没有选择列表了，这是因为 PC 无线模块是唯一可供连接的模块。如果你还有控制板直接通过 USB 线与电脑连接，就会出现选择列表，这里选择 PC 无线模块。当打开“interface test”窗口，出现错误信息——没有发现控制板。然而，这也无关紧要，因为你可以简单地对数据链路模块做出调整，当你改变 PC 无线模块频率并且按下“COM - USB”按钮，PC 无线模块和两个控制板又出现在列表中，因为现在三个模块频率再次相同。

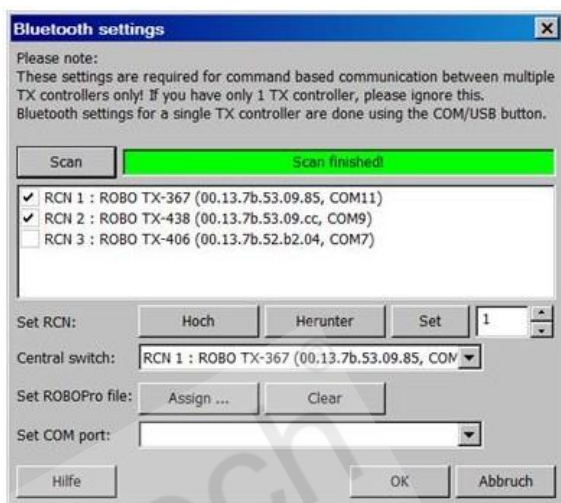
为什么你要首先改变控制板的频率，最后才改变 PC 无线模块的频率？如果你先改变 PC 无线模块的频率，试试会出现什么问题。

此外，会发生变化只有几秒钟的影响，如果“COM / USB”的选择列表不符合当前配置，只需再次按下“COM / USB”按钮。如果你还是不能通过无线连接控制板，最好直接用 USB 连接 PC 机和控制板，调整控制板的无线射频模块。



4.5.2 TXT 和 TX 控制板的蓝牙设置

每个 TXT 和 TX 控制板需要一个无线通信号码 (radio call number)，你可以在“蓝牙设置(Bluetooth setting)”窗口中，对每个控制板进行编号（1-8 之间的数字）。ROBO Pro 程序中，无线通信号码类似电话号码，用来识别各个控制板。首先，需要开启蓝牙功能，将所有控制板与电脑连接，TXT 和 TX 控制板的说明书中有详细描述。此外，你的电脑需要有集成蓝牙或外置 USB 蓝牙适配器，实现 TXT、TX 控制板和电脑信息交换。当所有的控制板打开并且连接到电脑后，打开 ROBO Pro 软件。按下“COM -USB”按钮，在在线模式下定义 TXT 或 TX 控制板的连接方式：通过 USB 或蓝牙连接。然后通过 ROBO Pro 菜单中**编辑蓝牙(Edit Bluetooth)**选项或使用工具栏上的蓝牙按钮，打开“Bluetooth setting”窗口，点击“Scan”，搜索 TXT 或 TX 控制板。



根据您使用的蓝牙适配器类型，检索列表中可能没有出现控制板信息，在这种情况下，你可以点击“COM / USB”按钮，选择相应的 COM 接口，通过控制板测试找出各个 TX 控制板。你也需要通过“**Set COM Port**”手动给每个 TXT、TX 控制板分配相应的 COM 端口。但在大多数情况下，这是自动进行的。

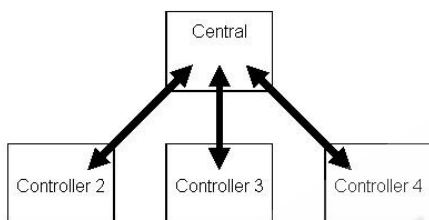
检测到的 TX 控制板列表总是根据无线通信号码(RCN)排序。你可以通过选择列表中“UP”和“DOWN”改变无线通信号码。例如，如果你选择的无线通信号码为 2，按下“UP”，编号变为 1，相反编号变为 3，同样可以使用“SET”进行编号设置，在“SET”按钮旁输入无线编号，并按下“SET”。

一旦你调整了无线电信号编号，你需要设定哪个控制板用于无线传输。作为一项规则，将是列表中的所有控制板。然而，在学校中，其他

小组的控制板可能出现在列表中。选择通信控制板时，在控制板前的方框内勾上“对勾”。

由于多次下载同一个程序到多个 TX 控制板中是十分繁琐的，ROBO Pro 软件提供了对列表中的每个 TX 控制板下载同一个程序的功能。请使用“Assign”和“Clear”按钮，“Download”窗口提供一次下载所有程序的选项。

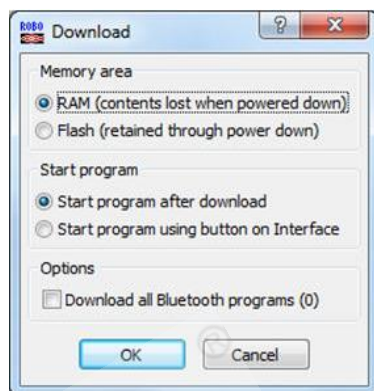
4.5.2.1 中央控制



3 个或更多的控制板之间的消息交换使用星型拓扑结构，其中一个控制板作为中央控制板，如下图所示。控制板 2 与控制板 3 交换信息，信息要通过中央控制板。通常选定哪个控制板作为中央控制板并不重要，但是，在移动机器人控制过程中，对于无线电范围之外的机器人控制，哪个控制板作为中央控制板就很关键，这同样用“Bluetooth setting”窗口进行设定。另外很重要的一点，运用中央控制的在线连接更加复杂，同时更加不可靠，这是因为连接电脑后，PC 机作为在线连接的中央控制板，同时中央控制板只作为蓝牙连接的中央控制板，如前所述，这样做同样起作用，但是需要更长的时间建立一个连接。

4.5.2.2 设置都保存在哪里？

控制板选择（通过勾选控制板前方的方框）下载程序到控制板是 ROBO Pro 程序的一部分。要进行多个下载时，所有的程序都默认蓝牙设置是开启下载。另外，无线信号和 TX 控制板分配不是保存在 ROBO Pro 程序中，而是保存在电脑上，这使得替换 ROBO Pro 程序变得更加简单。



3 个或更多的控制板之间的消息交换使用星型拓扑结构，其中一个控制板作为中央控制板，如下图所示。控制板 2 与控制板 3 交换信息，信息要通过中央控制板。

对无需讲解就可以管理蓝牙的专家们，蓝牙配对代码为 1234。

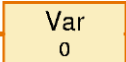
5 级别 3: 变量, 面板和指令

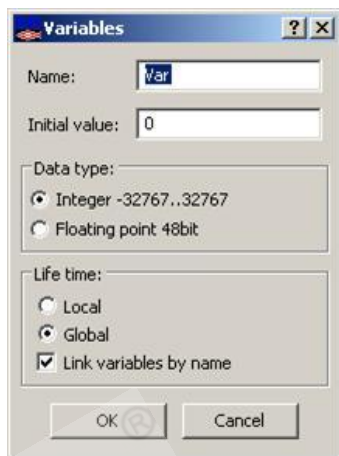
考虑用级别菜单将 ROBO Pro 重设到级别 3 或更高!

想象一下, 如果你在博物馆的过道里发现一台神奇的机器, 而且必须要用慧鱼构件仿制一台。而在你研究这台机器时忘记了时间, 没有注意到其他的游客正在离开这个博物馆。当博物馆关门后, 你可以对这台机器做一个彻底的研究, 并且能够做出一个复制品。但是, 你不得不先在博物馆中呆一个无聊的夜晚。要使这种事情不再发生, 你可以志愿到馆长那里编写一个计数器, 用它对进入博物馆和出博物馆的游客进行计数。并且只要仍有游客在博物馆里就打开红色的告警灯。但是, 该怎么做呢? 怎样用 ROBO Pro 进行计数呢? 答案是用**变量**。

Cedutech

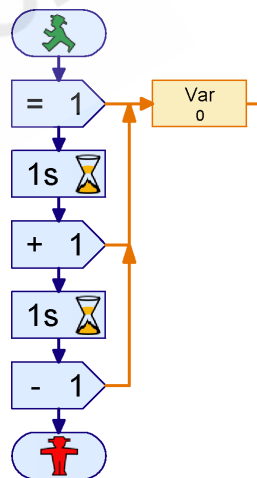
5.1 变量和指令

 一个变量是一种能够保存数值的模块。你可以在变量的属性窗口中输入一个**变量名(Name)**，它可以给你一些关于变量中存放何种类型数字的一些提示。在**初值(Initial value)**中你可以指定在程序开始时变量中存放的值。数据类型(Data type)决定变量是整数（例如 1,2,3）或浮点数（例如 1.2345）。现在，先使用整数。“Life time”设置将在 [8.4.1 全局变量](#)和 [8.4.2 局部变量](#)章节中详细介绍。



你可以通过发送指令到变量，来改变变量中存储的值。一个变量可以接受三种不同的指令：**=**、**+**和**-**。**=**指令是用新值覆盖存储的值。**+**和**-**指令是对存储的值做加减值运算。

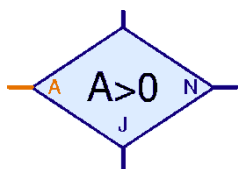
你可以用“**指令模块 (Command element)**”向变量发送指令。像其他的程序模块一样，指令模块的上面有一个蓝色的程序入口和下面一个蓝色程序出口。但是在右边有一个全新的东西，一个**橙色的**接口。那是指令输出。不管什么时候，指令模块执行时，它会通过这个输出向和它相连的模块发送指令。在左边，这个变量有一个相应的指令输入。当你把指令的输入和输出连接起来，ROBO Pro 会显示一条橙色的线条，而不是通常的蓝色的连接线。程序模块可以通过这些橙色的连接线发送指令，并且可以交换消息。



右边程序中，开始时向变量**Var**发送指令**=1**。这个指令包含了实际的指令**=**，和值**1**。**=1**指令将变量设置为1。1秒钟之后，程序向变量发送了**+1**指令，这个变量就把1加到原来的值上，现在得到2。又过了1秒钟，程序发送了指令**-1**，因此变量值又变为1。

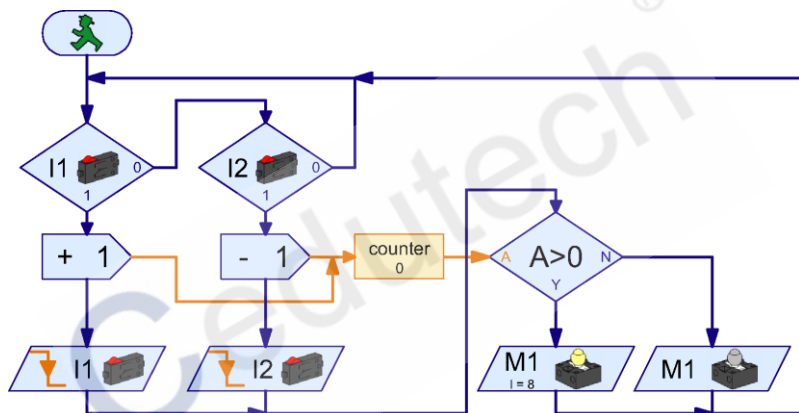
现在，尝试一下用 ROBO Pro 编一个简单的程序。你可以在“**指令 (Commands)**”组中找到指令模块，在“**变量，定时器.....(Variable,Timer,...)**”

组找到变量(Variable)。如果以在线模式执行程序，你会看到变量的值是如何变化的。



你可能会问：我可以看到变量的值，但是我能用它做什么呢？非常简单：在变量右边有一个橙色的端子，这个端子用于变量发送它的当前值到所有与变量相连的模块去的。ROBO Pro 中，有一些带左边橙色输入的模块，你可以把它和变量的输出连接起来。所以，例如，在“分支，等待，.....(Branch,Wait,...)”组中，你会发现 Yes / No 分支判断模块，它并不直接查询输入，而是可以在变量值中要求任意值。

所以，博物馆的游客计数器程序编写如下所示：



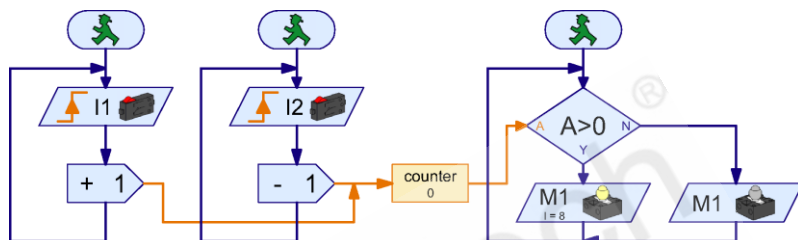
旋转门的入口有一个传感器 I1，旋转门的出口有一个传感器 I2。当 I1 压下时，程序发送一个+1 的指令到变量 Counter。然后程序等待，一直到 I1 再次被释放为止。对于在出口处的 I2 传感器，它的工作原理是完全一致，除了在这里用的是一个-1 指令将值发送到变量 Counter 中。每次计数器的值发生变化时，会检查计数器的状态。如果计数器 Counter 的值>0，红色告警灯会被打开，否则它会被关掉。



把上述程序做一次拷贝，并试一下。当你按下传感器 I1 并释放一次，告警灯 M1 马上会点亮。如果你动一下传感器 I2，告警灯马上熄灭。如果你按下并释放 I1 多次，你必须也要按下并释放传感器 I2 相同次数才能使得告警灯熄灭。现在，试一下如果先进来 5 个游客，然后出去 2 个，然后又有 3 个进来。那么，你要按 I2 多少次才能将告警灯熄灭？

5.2 变量组和多个进程

在你测试时，也许你已经发现如果同时按下 I1 和 I2 会出现错误。只要有一个传感器压下后，程序就不能响应其他的传感器了。由于，游客可能进入和走出专门正好是同一时间，这样就会产生计数错误。我们可以用几个并行的进程以避免这个错误。到目前为止，所有的程序都只有一个开始模块，但是这并不影响你使用多个开始模块。所有带有开始模块的程序都是并行运行的。所以专家们称之为并行进程。你可以用这项技术修改这个游客计数器程序，如下所示：



现在，两个互不相关的进程分别用于 I1 和 I2。如果传感器 I1 被按下，这对用于 I2 的进程是没有影响的，并且可以继续监控传感器 I2。另外还有一个独立的进程用于查询计数器的值和负责告警灯的打开和关闭。



正如你所看到的，若干个进程访问一个变量是没有问题的。你可以从多个进程向变量发送指令而且可以在多个进程中使用这个变量。变量是非常适合在进程之间交换信息的。

这个博物馆馆长会非常喜欢你这个智能的游客计数器，而且他会要求你解决另外一个问题：博物馆有一个新的展览。由于所有的游客都想看这个新的展览，以至于展览拥挤不堪，大家都看不到什么东西。所以，馆长想要限制该展览区域的游客数为 10 个。馆长在这个展览的进口和出口分别安装了旋转门。且旋转门可以以电子控制的方式被锁住。现在，他仅仅需要一个可以胜任的程序开发商：你！

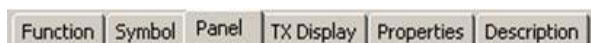


试着用 ROBO Pro 去开发一个上面提到的程序。它的功能基本上和游客计数器相同。你可以把红色告警灯 M1 看作电控锁，它应该在该展览区域有 10 个人时自动锁住门。

5.3 面板

当你解决了这个展览的问题后，博物馆馆长还有一个另外的任务交给你。它想知道一天内有多少游客访问这个博物馆。当然写一个计数的程序对你来说是没有问题的。但是，你怎么显示这个值呢？当然，你可以以在线的方式执行这个程序，跟踪这个变量的值。但是对于一个电脑盲就象馆长，那样就太复杂了。我们需要更简单一点的东西！

对于这种情况，ROBO Pro 里有面板。面板是一个你自己的可以放置显示和控制按钮的页面。装载你的游客计数程序并在功能栏上切换到**面板(Panel)**。



最初，控制面板是一块空的灰色的区域。在这块区域上，你可以放置显示部分和控制模块，它们在面板模块下的模块组中。在面板模块中你可以找到按钮，滑块控制等模块。在显示部分下你可以找到文本显示，指示灯，带旋转指针的显示等模块。



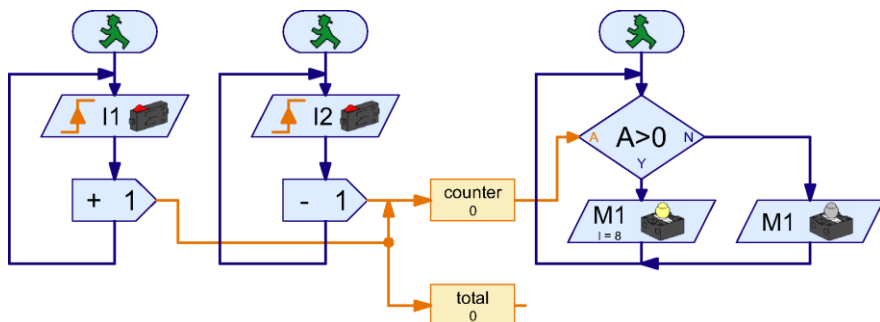
警告：面板是一个子程序的一部分。如果你有多个子程序，确认一下你是否是在主程序下创建了这个面板，而不是在一个子程序下。还有，作为一个高手，你可以创建多个面板。

如果你画了一个面板，后来突然消失了，原因应该是你在子程序栏上选择了一个子程序。那么，只要切换到主程序，你的面板就又会出现了。



对于游客计数器，你可以从 **Panel elements / displays** 模块窗口选择一个**文本显示(Text display)**（颜色不是问题）。并定好它在面板中的位置。这个显示板就可以显示博物馆游客的数量。

但是首先，你必须在你的程序中添加第二个变量，用它记录从入口进入的游客的数量，这个变量值是不需要减去出口出去游客的值的。在功能栏中，切换到**功能(Functions)**，插入变量 **Total**。如下所示：



正如你所见的，我们也可以用指令模块同时向两个变量发送指令。变量 **Total** 不接收 **-1** 指令，因为，指令只沿着橙色的连接线按箭头方向发送。另一面，**+1** 指令会发送到两个变量。这里只是一个例子。使用第二个指令模块会更简单更透明。

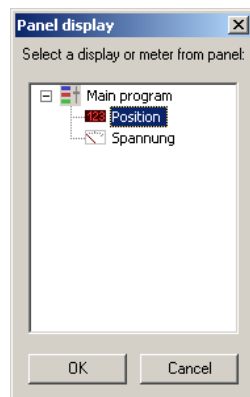
小贴士：如果给橙色连接线画分支，那么从它们的源头画到它们的终点通常会更有效。例如，你想画一根线到变量 **Total**，先点击变量 **Total** 的输入端，然后向后移动到分支点处。如果，在另一面，你想从已有的橙色线开始画一根橙色线，你必须双击（用鼠标左键）新线开始的点。



现在，你有了一个文本显示板和一个你想在显示板中显示的变量。接下来，我们怎么把这两个连起来？由于显示板和变量属于两个页面，你在连接这两个对象时会遇到麻烦。因此，软件提供了一个特殊的模块，它专门用于传送显示值到其相应的显示板去。你可以找到如上所述模块“**面板显示(Panel output)**”，它在“**输入，输出(Inputs, outputs)**”组中的最后一个。把一个面板显示模块插入到你的程序中去，放在变量 **Total** 的旁边。并将面板显示模块和变量连接起来。

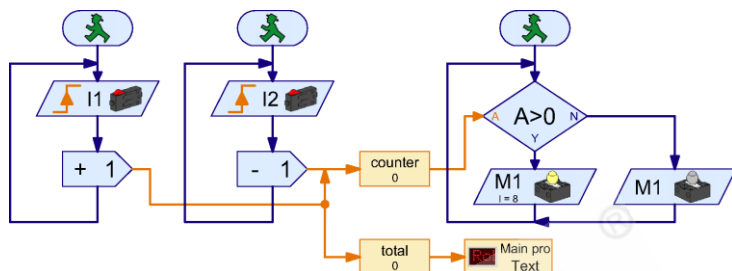


由于我们通常有不止一个显示板，你仍然需要让“面板显示”知道变量值发送到哪个面板上去。这可以通过在模块的属性窗口中设置。如果你用右键单击“面板显示”模块，你会看到所有到目前为止插入到面板中的显示板。由于每个子程序都可以有一个自己的面板，所以可以根据子程序列出所有的面板。在我们的例子中，有一个称为 **Text** 的显示板。选择这个显示板，点击 **OK**。



当你一连接“面板显示”到一个显示板，这个符号和标题就相应地变化了。我们使用的这个面板显示与（子）主程序中的 **Text** 的文本显示板建立了连接。

当你一插入面板显示，并把它和文本显示板连接起来，这个程序就成为如下图：



直接试一下。当你以在线的方式运行程序时，这个显示板就会显示进入旋转门的游客数。

小贴士：如果你想在一个面板上使用多个显示板，要给每个显示板取不同的名字。这样，当你在程序中连接它们时才能正确区分。你可以在 **ID / Name** 下输入名称。如果你想连接“面板显示”到这个显示板，这个名称就会显示在“面板显示”的选择窗口中。我们现在只有一个显示板，所以名称并不重要，我们保持它的名称为 **Text**。

这个程序还不是非常完美。现在还缺的东西是一个可以复位计数器的开关。然而，为了实现这个功能，我们不想用一个普通的按钮开关，用一个可以在面板上按的按钮。

Button

你可以在“**操作模块/控制(Operating / Control)**”组下的模块窗口中找到这个操作按钮。在这个功能栏上，切到“**面板(Panel)**”并在面板上插入按钮，插在文本显示板旁。给它取名称为 **Button** 是非常合适的，名称很容易在按钮的属性窗口中更改的。输入 0000 作为**描述**，并以 **OK** 确认。

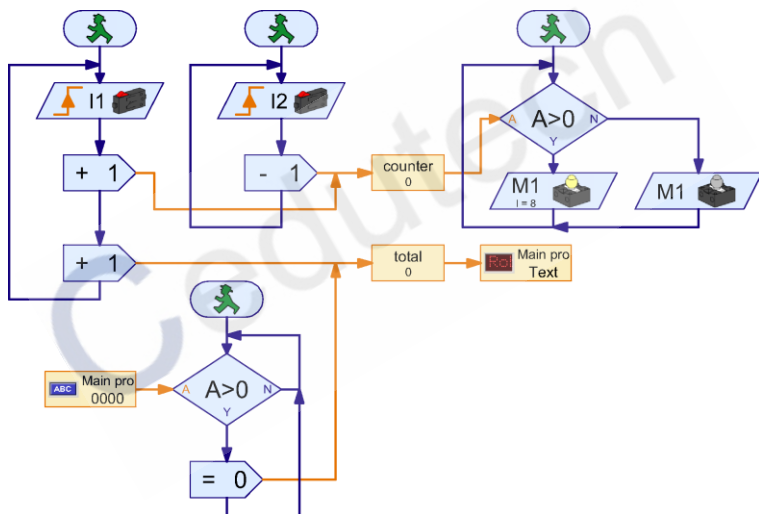
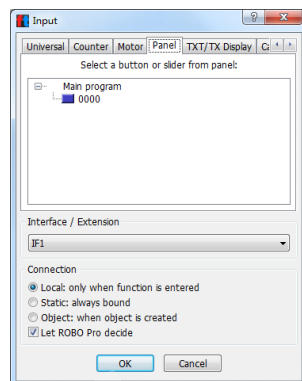


就像文本显示板一样，我们也需要一个模块将按钮连接到程序。切换回功能栏上的“**功能(Function)**”窗口。可以在模块窗口的**输入输出(Inputs, outputs)**组找到图示的 **Panel input** 模块。把它放在流程图中，现有程序下方。

现在你仍然要在面板中将输入面板和按钮连接起来。你可以通过对输入模块右键单击完成连接。和显示板类似，控制模块也是根据子序列出来的，因为每个子程序可以有自己的面板。现在，选择按钮 **0000** 并按 **OK** 确认。

你可能注意到通过属性窗口的标签栏，可以从这个模块切换到其它各种输入的设置。然而，这会在下面的章节中介绍。

面板输入传递的值会被判断模块查询。你实际已经使用过这个模块去查询变量。这个完整的带有“清零”功能的程序如下所示：



只要按钮 **0000** 按下去，“=0”指令会发送到 **Total** 计数器，并将计数器置零。

5.4 定时器

在你胜利完工之后，博物馆馆长终于知道了该怎么做，并任命你为博物馆的电脑顾问。当然，这是一个有荣誉和声望的职位，但是同时也有很多工作。例如，下面提到的：博物馆有很多只要按一下按钮，会移动模型。但是，一些游客长时间按着按钮，所以模型过热了，并一直要送出去修理。现在，馆长想让模型在按钮按下去时运行，但是一次最多运行 30 秒。在模型运行一次后，应该在再次运行前休息 15 秒。

嗯！没问题，你可能在想。一会儿，你就有了一些程序。试一下吧！但过后，你会感觉到这个不那么简单，有两个原因：

- 在 30 秒内，程序必须查询这个按钮看它是否在 30 秒之前释放了。好，你可以用两个并行的进程解决这个问题。参考 [5.2 章节“变量和多个进程”](#)。
- 如果游客在 5 秒后放了按钮并且过了 15 秒后又按下了它，那么 30 秒的时间延时器必须重新来过。但是时间只延时了 $5 + 15 = 20$ 秒，所以还是处于激活状态。即使用并行进程，你还是不能让延时重新来过。也许，可以在三个进程中使用两个延时交替启动解决这个问题，但是这样会带来很多麻烦。

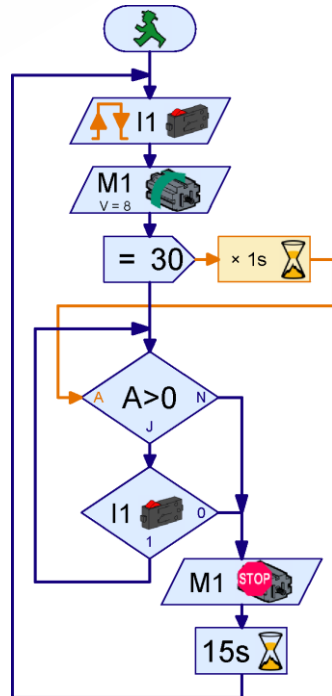


就没有更简单的方法吗？有的，称为**定时器变量(timer variables)**，或简称**定时器(timers)**。最初，定时器就像一个普通变量，这个定时器跟踪数值，并且你可以用 $=$ 、 $+$ 和 $-$ 指令改变数值。然而，定时器的特点是它会以固定的时间向下计数一直到零为止。数值减少的时间间隔可以以千分之一秒到一分钟为单位进行设置。许多时间控制问题用定时器比用时间延时器会更容易解决。知道如何用定时器来解决问题了吗？

正确答案：当游客按下按钮 **I1** 时，你启动模型并且设置定时器，使用 $=$ 指令， $30 \times 1 \text{ 秒} = 30 \text{ 秒}$ 。然后进入循环，检查 30 秒的时间是否到了或 **I1** 是否被释放了。当两个中任何一个停止条件满足，停下模型并等待 15 秒。然后从开始端重新开始。



正确





必须承认，我们对程序有更多的要求了。试一下，解决下面练习中的问题：开发一个相同的程序但是使用时间延时器而不是定时器。**注意：这个练习是非常难的。只是为了那些特别喜欢解决难题的人设计的。其他人可以直接进入下一节。**有两个办法解决这个问题，你可以使用两个时间延时器，它们在它们自己的进程中交替运行。由于这里有 15 秒的停顿时间，其中一个时间延时器在第二个周期的末尾计时时间到，可以重新启动。另一个方法是用普通变量和一秒短延时的**时间延时器**模拟一个定时器。

Cedutech®

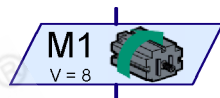
5.5 子程序指令输入

通常，博物馆的模型配备了控制板，且你的程序工作得非常好，慧鱼公司很欣慰。但是，像其他的公共机构一样，博物馆也受资金的困扰。所以，馆长想要用尽可能少的控制板。一个控制板有四个电机输出和足够的输入去控制四个模型。由于大部分电机只向一个方向旋转，你可以通过单级输出 O1 到 O8 控制最多 8 个电机。

这样可以为馆长省下很多资金。但是，你必须把程序拷贝 7 次并要调整所有的输入和输出。或者也许不需要？你不能用子程序来实现吗？

实际上，是可以的。当这里有一个问题出现了：

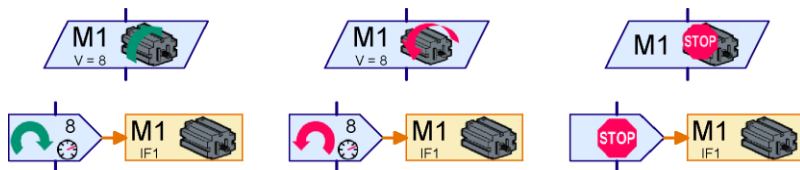
如果你在子程序中使用**基础模块(Basic elements)**组中的普通的传感器查询的话，每次子程序的调用会查询相同的传感器和控制相同的电机。原因是，例如在电机输出模块用于控制电机的控制指令和电机的输出号码是同一个模块。由于，只有一个版本的子程序，所以总是出现相同的电机。如果你改变一个子程序调用的电机号码，对于所有调用该子程序中的电机号码都会改变的。所以，你又不得不拷贝子程序 7 次，给每个子程序一个不同的名称，且手工调整所有的输入和输出。



但是，有一个更有效的解决问题的方案。方法是把电机和控制指令分开。然后你可以把控制指令（向右，向左，停止）放入子程序中和将电机模块放在主程序中。



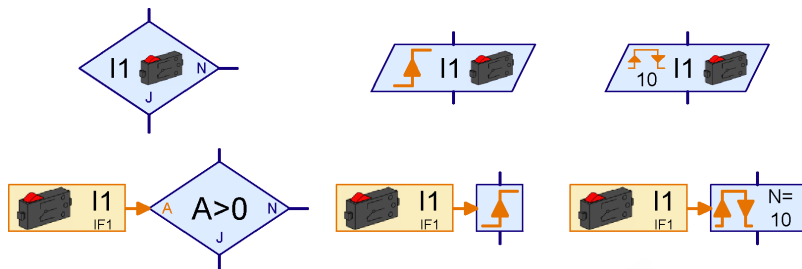
在子程序中，使用指令模块，这是你在变量中遇到过的，然后，你向主程序发送向右，向左，停止的指令，你可以把它们分配到各种电机中。对于电机，有一个专门代表电机的 **Motor** 模块，它不能确定电机怎么运行。这个模块有一个指令输入，可以将指令发送到这里。你可以从“**基本模块(Basic elements)**”组中用指令模块和电机模块代替这些模块。如下图所示：



在上面一排中，可以从“**基本模块(Basic elements)**”组中找到这些电机模块。第二行中，描述的是相应的组合，它们可以达到相同的作用。它们由“**指令集(Commands)**”组中的指令模块和“**输入输出**

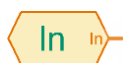
(Inputs,output)”组中的电机模块组成。实际上，上面的模块是下面的缩略和简化。每种都能向电机 **M1** 发送向右，向左，停止的指令。

这些也适用于查询传感器：



在上面一排中的模块，你可以从“基本模块(Basic elements)”组中找到。在下面一行中，是相应的数字量输入和在“判断，延时(Branch, Wait)”组中的各模块的组合。你可以在“输入输出(Inputs,output)”组中找到橙色的“数字量(Digital input)”模块。

使用这个办法，你可以把程序的逻辑从输入和输出中分开。当是仍然缺一些东西。如果电机和传感器放在主程序中且指令放在子程序中。你会发现需要一个子程序入口出口(Subprogram I/O)组中的连接模块。

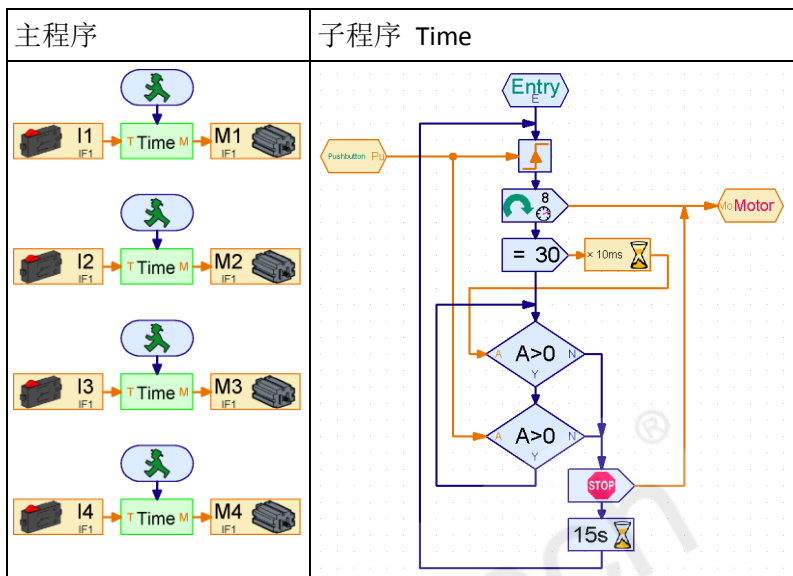


通过子程序的指令输入，你可以从外部向子程序发送一个指令。如果输入状态发生改变，数字量输入模块（传感器）通过橙色连接线发送它的新值（用所谓的“=指令”）。在模块对话框中你可以给输入定一个名称。



通过子程序指令输出，你可以从子程序发送指令。因此，例如，你可以从子程序向电机发送向右、向左、停止的指令。对于这个模块，你也可以在对话框中输入一个名称。

现在，你的带子程序的多定时器模型就绪了。



子程序 “Time”几乎和前面章节中的程序是一样的。在开始处和循环里的 **Wait for digital input** 模块被 **Wait for** 模块替代，使用了 **Branch, Wait** 组中的用于数据连接的橙色连线。它们都连接到了子程序指令输入 **Sensor**。在开始处和结尾处的两个电机控制模块被指令模块所替代。它们都向子程序指令输出 **Motor** 发送它们的指令。



子程序 **Time** 在主程序中被调用了四次。子程序指令输入 **Sensor** 自动生成绿色子程序符号的左边的橙色连接 **S**。右边连接端 **M** 是用来连接子程序指令输出 **Motor**。子程序符号的连接 **S** 分别连接了每一个传感器 **I1** 到 **I4**。 **M1** 到 **M4** 中每个电机分别连接了连接点 **M**。通过这种方式，每次调用子程序 **Time** 查询不同的传感器和控制不同的电机！

拷贝以上子程序和主程序并试一下。你必须先编一个子程序，因为否则你就不能把子程序插入到主程序中。如果你有什么子程序的问题，再次请参考“[级别 2：用子程序控制](#)”。

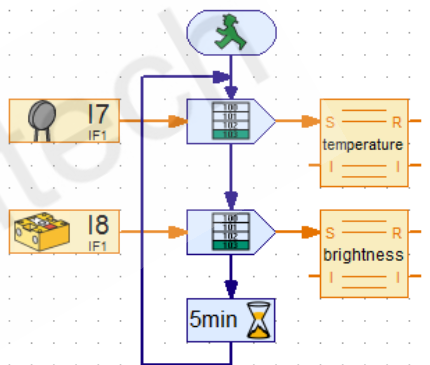
小贴士：你将在 [6.3 章节“向子程序发送自定义程序”](#) 中了解到更多关于指令输入的信息。

5.6 列表（阵列）

现在，通过使用了你的经济的控制系统，博物馆的所有试验设备都解决了。没多久馆长有了下一个问题：在某个区域放了一些非常昂贵的古董展品，最近那里出现温度的变化，而这对古董是有害的。你认为这和光照有关。为了，说明其中的关系，你先做一个设备，它能记录光照程度和温度值。当然，控制板通用输入端可以接收模拟量信号，你也知道如何利用变量存放值。所以，整个事情应该没问题，是这样吗？每五分钟记录两个值，12 小时需要 288 个变量。那样会出现一个非常巨大的程序。我们能用于程序来简化它吗？可以，但是这里有更好的办法，列表模块（程序员称之为阵列）。

你可以在列表种存放不止一个数值而是一串数值。最初，列表是空的。如果你发送“**添加(Append)**”指令到左上方标志为 **S** 的数据输入模块中，指令中指定的值会添加到列表的末端。你可以通过**列表(List)**模块的属性窗口设置列表的最大长度为 1 到 32767。这样，就很容易记录光照和温度：

将温度传感器接在 17 处，并将亮度传感器接在 18 处，程序以循环的方式每 5 分钟读取这两个值，并把它们用 **Append** 添加到列表中去。

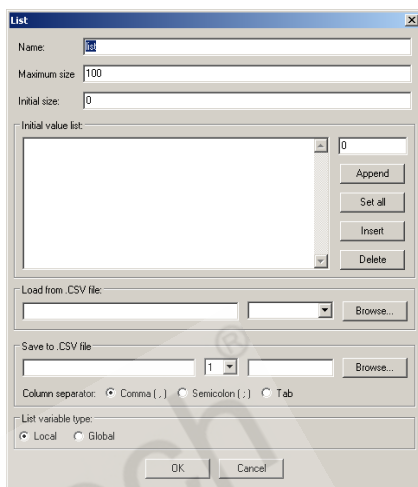


小贴士：当插入指令模块时，你必须激活属性窗口中的数据输入的选项。然后，数据输入就会出现在指令模块的左边，然后你可以把模拟量输入接上去。

为了便于测试，有必要将循环时间从 5 分钟改为几秒钟。

现在你一定在想如何才能从列表中将数据读回来呢。有两种方法：你可以像读取一个普通变量一样读取它，并在你的程序中处理它。由于列表中包含多个数据，你首先要选择你要读取的数据的数据编号，在左边的数据输入，标志为 **I**。然后，这个模块的值就会在右边的数据输出模块 **R** 中给出。

ROBO Pro 也可以将列表中的所有值保存到你的电脑中的文件内，比如 Excel，对它做进一步的处理。由于目前的情况下，你只想查看和比较所记录的光照程度和温度值的关系，所以，这样做毫无疑问更可行一些。ROBO Pro 把数值保存到 **CSV 文件** 中。CSV 文件是一种每一系列数据包含在一列或多列中的文本文件。因此，你可以将温度和光照的测量值保存在 CSV 文件的不同的列中。列与列用逗号隔开。在一些国家中 0,5 用逗号隔开而不是用句号隔开 0.5，（如德国），所以通常会使用分号（；）作为列分隔符。如果你在 ROBO Pro 和微软 Excel 之间交换 CSV 文件有问题的话，你可以在列表的属性窗口中修改列分隔符。



用“**存入 CSV file (Save to CSV file)**”，你可以在列表的属性窗口设定一个 CSV 的文件名和要保存下来的列名。程序以在线模式后结束，或者程序还在运行时（在线或下载模式）在**文件(File)**菜单下选择了 **Save CSV file** 项，数据可以保存下来。在下载模式下，把接口板和电脑分离来记录数据，也可以重新连上电脑保存数据。

当你在在线模式下执行了上述的程序后，你可以在微软的 Excel 中或在其他的制表软件中打开 ROBO Pro 中记录的数据 CSV 文件。如果你没有制表软件，你也可以使用 Windows 编辑器(notepad)，通常你可以在 windows 的开始(start)菜单中附件(Accessories)子菜单中找到它。

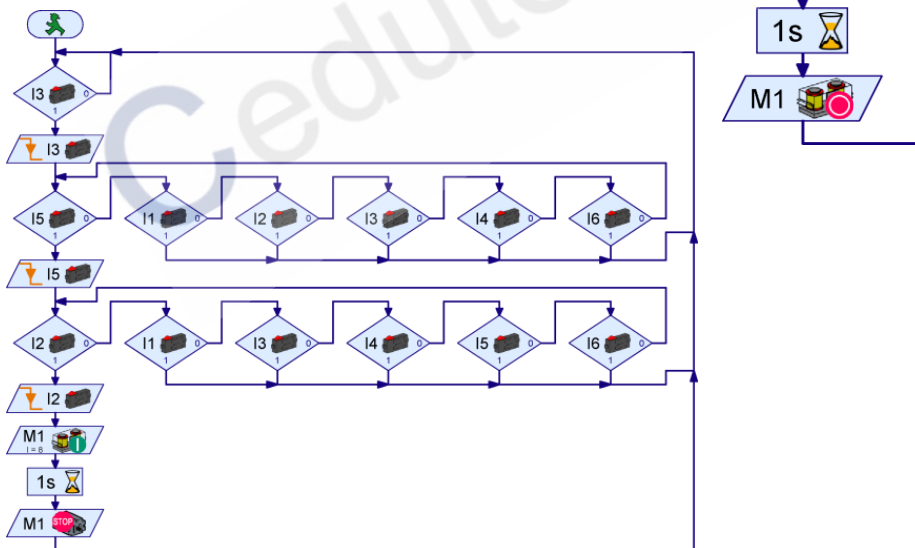
只要程序还在在线方式运行，你可以通过右击 List 模块查看列表中的数据。

5.7 运算符

光照和温度记录程序运行良好，但是很明显从记录的数据中可以看到展览空间内的温度和光照没有关系。这说明了一些游客把模型控制板和空调控制板搞混淆了。展览空间内的温度变得很糟糕就不奇怪了。

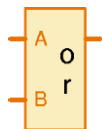
但是，通过电子密码锁，这个问题可以很容易避免。电子密码锁是一个含有按键 1 到 6 的键盘。如果三个值一个接一个的被正确输入，密码锁会通过磁铁释放温度控制板的盖。

起初认为，这个锁是相当简单的：程序只要等待有正确的键并且以正确的顺序按下。我们看到 3-5-2 组合的程序如右图所示。但是，更进一步的测试，发现这个程序有一个问题。这个锁很容易打开，你只要按顺序按下 1 到 6 三次。在任何情况下，都可以用此方式来找到并按下正确的键。爱因斯坦说得太恰当了：“每件事情都应该尽量简化，但不要过度简化。”所以程序不仅要查询是否按下了正确的键，还要查询是否有错误的键按下。现在，改进的程序应该如下图：

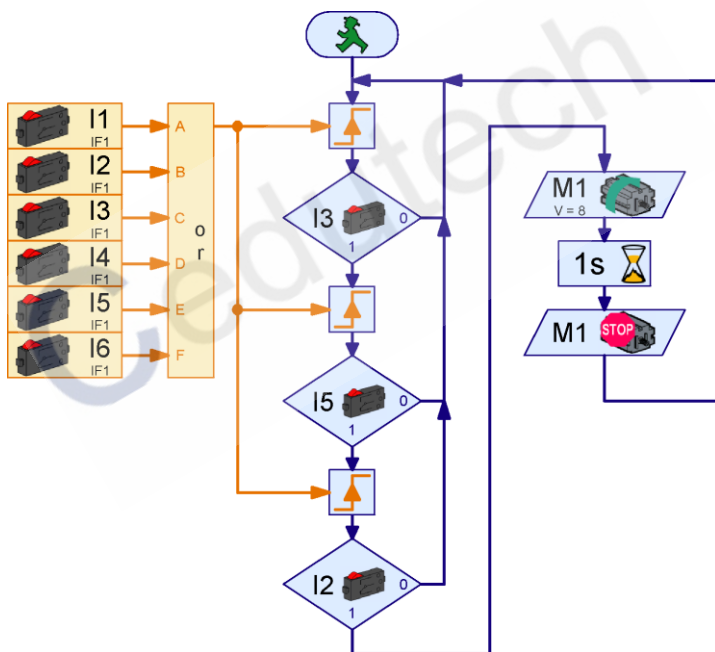


这个程序只有当键 3-5-2 按下并且期间没有其它的键按下时，锁才会打开。比如，如果键 3 按下了，程序先会等该键释放。如果，除了 5 之外得其他键按下，程序会重新开始。所以，程序能正确工作，但是

它既不够简单也不够突出。而且，也很难改变密码。但是不用担心，可以用运算符正确而又简便地完成这个任务。这里有各种各样的运算符。可以在“**运算符(Operators)**”组中的程序模块中找到它们。对于这个密码锁，我们首先需要“**(OR)或**”运算符。



若干个信号可以接入到**运算符 OR**上，只要输入信号中至少有一个 1，它就会产生 1。如果几个按钮传感器接到**运算符 OR**上，只要有一个按钮被按下该运算符的输出就为 1。输入的数量可以在运算符的属性窗口中设置，最多可以到 26 个。所以，可以将所有 6 个按钮接到一个运算符上。也许，你会问我们如何用它来简化这个密码锁？非常简单：通过运算符，一开始你可以等待，一直到有键被按下，然后，你可以检查键是否正确。接下来，对于每个数字，你只需要 2 个而不是 7 个程序模块。



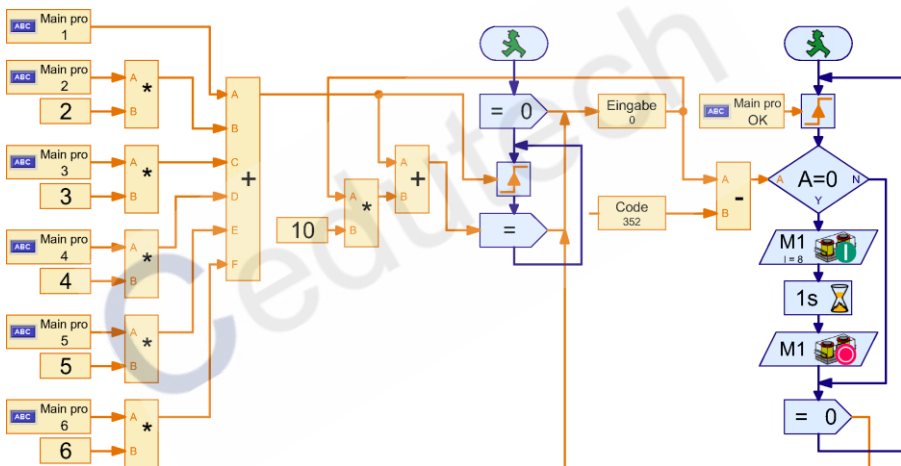
I1 到 I6 这些输入按钮一起连接到有 6 路输入的“或”运算符 OR 上。如果只要其中有一个按钮按下，“或”运算符 OR 产生 1，否则是 0。通过 **Wait for** 模块，程序等待一直到有一个按钮按下。顺着这个，我们马上测试它是否是一个正确的按钮。如果是的，我们等待下一个键的输入，如果不是，程序从头开始。

修改上述程序，在面板中使用面板模块代替按钮传感器。画一个带有 6 个标号为 1 到 6 按钮的面板。然后在属性菜单中修改数字量输入。你必须用数据输入和分支输入来逐个代替原有的判断模块。



密码锁现在没有问题，但是修改密码(352)仍然不是很容易。我们必须修改三个判断模块中的输入。没有必要经常修改博物馆的空调系统的密码，但是，例如，将密码锁用于一个报警系统，可能要经常改变密码。当然，如果密码保存在变量里会更加容易。密码甚至可以自动修改。例如，如果我们要在报警系统中出现一个静音报警，可以用特殊的报警密码代替普通的报警密码。

为了比较带有输入的组合变量，你必须把输入本身存放到变量中。在开始时，输入变量的值是 0。当你按键 3，变量的值应该是 3，如果再按下 5，则得到 35，最后按下 2，得到变量值为 352。




带密码变量电子锁有两个进程。在左边的进程中，用乘法和加法运算符给每个键赋一个数值。键 1 得到 1，键 2 得到 2，等等。键返回值 0 或 1，且如果乘以一个固定的值 x ，得到 0 或 x 。未按下的键值为 0，你可以把所有的值加起来，得到一个数值。当一个键按下时，输入变量值为 10 乘以前一个值加上按下的键值。乘以 10 使得输入变量已有的值向左移动一个 10 进制位（如，35 变成 350）



面板上 OK 键按下去后，右边的进程紧接着执行。如果密码变量 Code 中的密码输入正确的话，其值为 352，将和输入变量的值作比较。如果两者相同，电磁铁就激活。反之则不会。最后输入变量的值复位至 0。

变量 **Entry** 和 **Code** 互相比较差值是否为 0。这里也可以使用一个比较模块。



如果，你同时按下两个键，这两个键的值相加。例如，如果你同时按下 3 和 6，会得到 9。通过这种方式，你可以做一个超级保密码锁，这个锁必须要求有时同时按下若干个键才能打开。考虑一下按哪些键以怎样的顺序才能打开密码为 495 的锁？不要忘记，当数值增加时，**Wait for** 模块使程序继续下去，而不仅当它从 0 变为 1 时候。

这个电子锁密码是否能用 2 到 4 个数字？如果可以，最多可以用多少个数字呢？为什么？其他的电子锁程序又是怎样的呢？

Cedutech®

6 级别 4：用户定义命令

请在**级别(Level)**菜单下，将级别改为**级别 4(Level 4)**或更高！

ROBO Pro 的级别 3 中你可以大量使用控制指令处理数据，例如可以控制电机，因此需要对命令提前设定速度、左转、右转和停止命令。在级别 4 编程中，你可以通过橙色连接线使用自定义命令实现以上功能。

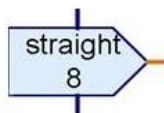
Cedutech®

6.1 进程命令

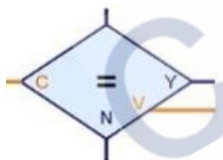
编辑好通过轮子或履带控制机器人小车运动的程序后，需要通过设定两个电机的转速和方向来实现机器人的左转、右转、前进和后退。同时你还要注意哪个电机驱动右边的车轮，哪个电机驱动左边的车轮，对于两个电机怎样的命令对应小车前进、后退和转向。但是对于聪明的头脑，是不能拘泥于小节的。

当然，你可以对每个电机运行模块编写子程序，但是如果编写一个通用的电机控制子程序，你只需要给该子程序发出前进、后退、转向等控制命令，电机就能做出相应的动作，是更加明智的选择。

你会发现 ROBO Pro 软件中没有前进或后退命令单元，这时你可以编写一个自定义命令符，属性窗口中注明“前进(straight)”命令后，这个命令符就可以实现前进功能。



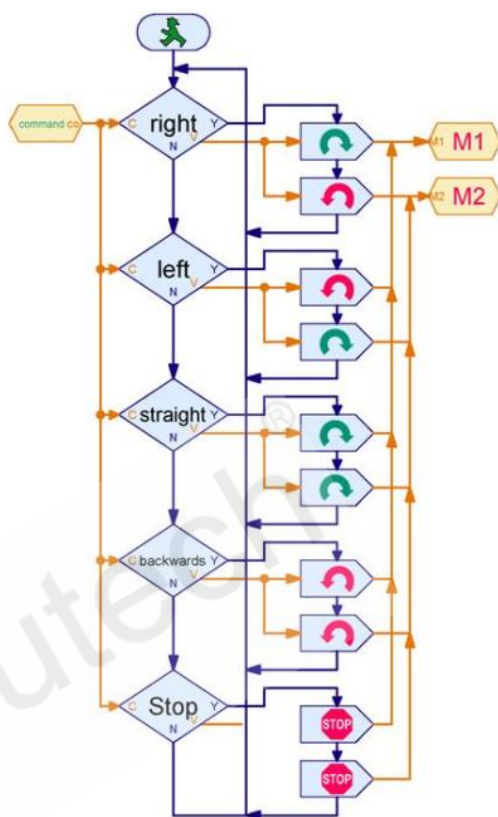
接下来我们怎样使用前进命令符呢？不管怎样，还没有指令能使用自定义命令符，如：你命令电机实现前进功能，这时 ROBO Pro 软件跳出“没有输入实现前进命令”。从级别 4 编程开始，在“接收，发送”命令单元下面出现“命令等待(Command Wait)”和“命令过滤器(Command filter)”命令符，这两个新命令符可以用来操作随机命令。



使用以上命令符扩充子程序，可以通过前进、后退、左转和右转命令控制双轮机器人。首先使用命令等待功能，可以将自定义命令符通过输入端 **C** 输入，调整该模块等待的指令。当等待命令符接收到指令后，程序在 **Y** 和 **N** 两端不断改变，检测命令符之后，命令符数值在 **V** 端输出。注意，在 ROBOPRO 软件中，命令包含其名称和数值。

编写子程序很简单，图示中循环程序使用了 5 个“等待命令”。当输入端接收到对应指令后，不同速度值经由控制模块传送给 M1 和 M2。如果子程序接收到的指令为“straight”，其值为 8，那么速度值 8 经由 straight 等待命令的 V 端口到达两个顺时针命令的输入端口，然后向电机发出指令。事实上，这里需要两个顺时针命令，应为后续要控制两个电机，特别针对两个电机控制方式不同的情况。

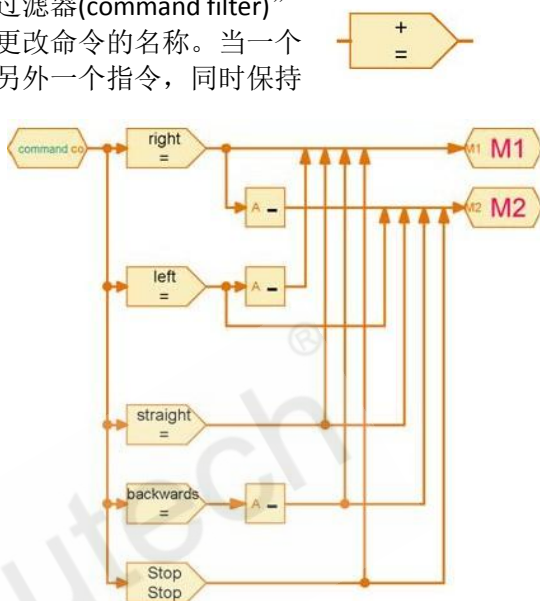
在例子中，“Stop”等待命令的 V 端口空置，因为停止电机无需数值。



6.2 命令过滤器

上述命令也可通过“命令过滤器(command filter)”来实现，使用命令过滤器可以更改命令的名称。当一个命令在左端输入后，右端输出另外一个指令，同时保持数值不变。这样，你可以做一个电机控制指令，把顺时针或逆时针命令变为赋值命令。使用命令过滤器，你可以将自定义指令转换为 ROBO Pro 的标准指令，就可以将自定义指令变为实际动作。

右图是使用命令过滤器控制双轮机器人的子程序，如：将改变“CW(right)”命令更改为赋值“=”命令，电机输出可以使用数值为-8 到 8 之间的赋值“=”命令来进行控制。在前进或后退时，保持小车的电机转向相同，由于小车顺时针转动需要两个电机有不同的转向，在使用赋值“=”命令后，对于 M2 电机，需要对数值取负，采用“-”运算符。小车逆时针转动时情况刚好相反。



对于停止(Stop)命令，其实无需使用命令过滤器，因为输入输出都为“Stop”命令。但是对于其他指令，比如 right, left 命令，不能直接传递给电机，所以这个停止命令还是有必要放在这里。

命令过滤器最大优点是节省程序进程，进而节省了存储空间，同时响应迅速。

应用命令过滤器时，需要注意，不要混淆各个电机的连线，如图所示，所有结束箭头都指向同一条数据线。有时对于一个指令使用两个命令过滤器会使程序变得更简单，因为这样就有两个独立的输出。



在上述程序中，“right”指令控制小车前进。尝试下将该指令改变为 M1 电机运行，M2 电机停止。你需要两个命令过滤器，一个将 right 转换为=，发送至 M1，另一个将 right 转换为 Stop，发送至 M2。



6.3 向子程序发送自定义指令

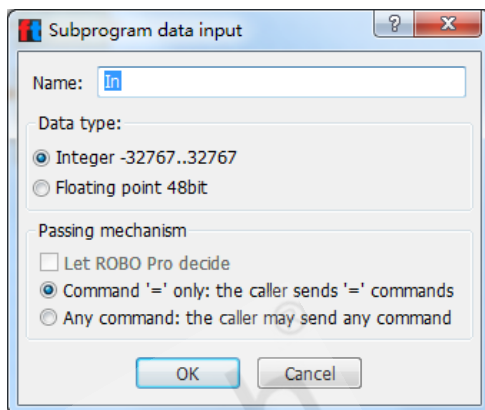
在 [5.5 子程序指令输入章节](#) 已经介绍子程序指令输入步骤，将数字量或模拟量模块作为指令，这些模块只是在输入改变时发送赋值“=”指令。如果向子程序输入端口发送其他指令，需要注意子程序数据输入端口

(Subprogram data input) 的属性窗口。在级别 4 开始，属性窗口中增添了“**传递机制(Passing mechanism)**”选项。

如果你选择“**Command** ‘=’ only”，只能将赋值“=”指令给对应的子程序输入端。另外，赋值“=”命令在子程序启动后自动重复进行，否则子程序输入值可能不正确。想象一下，一个数字量模块（比如微动开关）与子程序输入端连接，这些模块只在状态改变时才发出指令，子程序才会有输入。如果现在输入端闭合，发出为“1”的数字量指令。当子程序在指令发出后才被调用，命令持续发出是非常重要的。否则入口端会是错误数值，直到与其连接的模块状态再次改变。

但是这种自动传输模式干扰性很大，并且不被其他指令识别。如：向子程序发送“+1”指令，你不希望程序自动重复“+1”，这时可以选择“**任何指令(Any command)**”选项，以防重复发送指令。

注意：如果选择“**any command**”选项，向子程序输入端口发送“=”指令，子程序启动后指令却不重复，可能使子程序输入值与实际输入值不符。



7 控制多个控制板

一个 ROBO TX 控制板足够控制比较紧凑的模型，但是有些用户喜欢做复杂一些的模式。如果你的模型中超出了控制板上的输入输出端口数量，你可以通过 6 针扩展接口，扩展至多 8 个额外的 **ROBO TX 控制板**。对于仍然使用早期 ROBO 接口板的用户，可以在在线模式下，至多扩展 3 个 ROBO 接口板的扩展板。对于 ROBOTICS TXT 控制板，使用 10 针扩展接口，可以扩展 1 个控制板。

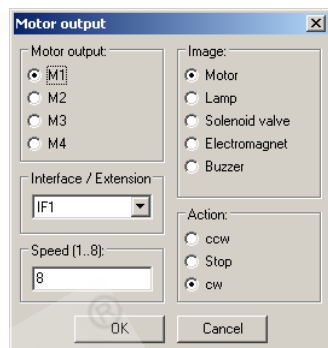
Cedutech®

7.1 扩展模块

或许你已经注意到位于输入和输出模块的属性窗口中，选项“接口板/扩展板(Interface / Extension)”下有下拉式菜单。可以选择你所要控制的输入或输出所在的接口板或者扩展板。假设你未做任何其它设置(参见下个部分)，列表中将有以下菜单：

- IF1: 只是与电脑连接的 ROBO TX 控制板，作为主控制板(master)。
- EM1..EM8: 只是与主控制板连接的 ROBO TX 控制板，作为扩展板(extensions)。

因而使用类似于 ROBO I/O extensions 扩展板是相当容易的。你只需要为输入和输出选择选择所需板子（接口板或扩展板 1-8）作就可以了。ROBO TXT 控制板的与 ROBO TX 控制板扩展方式相似。对控制板，如何设置为扩展板，详情请参考对应控制板的使用手册。



7.2 TXT 控制板, TX 控制板和 ROBO 接口板同时控制

如果你想用一个程序同时控制 ROBOTICS TXT 控制板, ROBO TX 控制板和 ROBO 接口板, 这只能在线模式下进行。你可以将带有 8 个扩展板 (ROBO TX 控制板作为扩展板) 的 ROBO TX 控制板通过 USB 数据线与电脑连接。另外将 ROBO 接口板通过 USB 或 COM 端口与电脑连接, 可以有至多 3 个扩展板。如果还是不够, 再将 ROBOTICS TXT 控制板与电脑连接, 可以有一个扩展板。为了在输入与输出模块的属性窗口中找到各个接口板, 必须对接口板进行配置。

在你没有更改设置的情况下, 你可以在下拉式菜单 (Interface / Extension) 中找到列表 IF1, EM1-EM8。也可以增减和修改这个列表, 为了以下几个理由:

- 为了好的可理解性, 可以为每个控制板命名, 而不是简单的称其为 IF1 或是 EM1。可以根据控制板控制机器或机器人的不同部分进行命名。
- 你可能想交换两块扩展板 (例如 EM1, EM2), 减轻换线负担, 不改变程序。
- 可能你要运行程序, 原来是基于 ROBO TX 控制板写的, 并且程序中涉及到多于 3 个扩展板, 现在想在多个 ROBO 接口板上使用。

在主程序 (main program) 的属性 (Properties) 窗口中, 你可以非常方便地通过改变控制板的分配 (Interface allocation) 来改变。

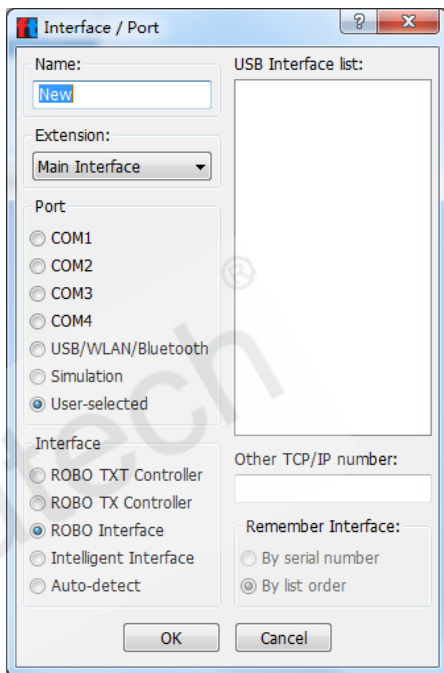
The screenshot shows the 'Main program' properties window in ROBO Pro. The 'Interface allocation' section lists the following interfaces:

- IF1 = Main Interface
- EM1 = Extension 1
- EM2 = Extension 2
- EM3 = Extension 3
- EM4 = Extension 4
- EM5 = Extension 5
- EM6 = Extension 6
- EM7 = Extension 7
- EM8 = Extension 8

The 'Symbol generation' section has 'Automatic' selected. The 'Default placement' section has 'Dynamic' selected. The 'Minimum number of processes' is set to 5, 'Additional processes' is set to 0, 'Min. memory per process (download)' is set to 8192, and 'Min. memory per process (online)' is set to 65536.

这里，可以看到已被命名为 **IF1** 到 **EM 8** 的各个模块（主控板或扩展板）。通过“**新建 NEW**”按钮，可以增加一个控制板。假如你希望编辑已在列表中的选项，可以单击“**编辑 EDIT**”，随之显示一个窗口，如右图所示。

- 在“**名称(Name)**”一栏，可以修改控制板的名称。名称不应该太长，因为图示符号中预留接口板名称的空间非常小。如果你更改了这个名称，必须更改使用这个名称的所有输入和输出模块中的控制板名称。
- 在“**扩展板(Extension)**”一栏，可以把这个名称归类为一个控制板还是一个扩展控制板 1 到 8。
- 在“**端口(Port)**”一栏，你可以选择为控制板分配的端口。如果你选择“**用户选择 (User selection)**”，所使用的端口将是你工具栏 **COM/USB** 选择项中所选。只要你想只用一个 ROBO TX 控制板带多个扩展板，就简单了，因为这种方法其他使用者无需更改。如果电脑还通过 USB 数据线连接有 ROBO 接口板，就要在此定义各个控制板所连接的端口。
- 在“**控制板(Interface)**”一栏，可以选择使用何种接口。如果通过串口连接有早期的 ROBO 接口板或智能接口板，程序能自动检测接口板的类型（“**自动检测(Automatic)**”选项）。
- 只有当你连接了多个控制板到 USB 总线，窗口的右边部分才会有用。如果在 **Port** 栏下，选择了 **USB**，可以选择 **USB Interface** 列表下的一个端口。



警告：与 ROBO 接口板不同，ROBO TX 控制板可以通过 USB 数据线或蓝牙连接到电脑，同时作为主控制板的 ROBO TX 控制板可以连接至多 8 个作为扩展板的 ROBO TX 控制板。

如果你打算在 USB 总线上运行多个控制板话，必须首先给各个控制板分配各自的序列号。缺省状态下，所有的接口都被分配了共同的序列号，以避免更换控制板时产生问题。在 Windows 操作系统下，只能检测不同序列号的控制板。你将在[第 7.5 节“改变控制板序列号”](#)中学到更多的相关知识。

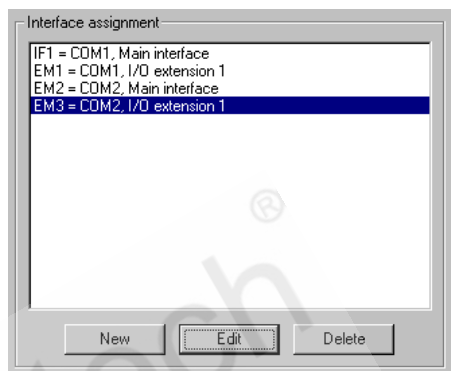
- 在“**记住控制板(Remember Interface)**”一栏，可以设定让程序如何记住所选择的接口板。这里有两个方法：如果选择了“**按序列号(By serial number)**”，控制板的序列号储存在程序中。如果把其它控制板连接到 USB 总线并移除，程序总是可以通过序列号找到所选择的接口板。换句话说，这样有个缺点，那就是程序必须和相同序列号的控制板来配合工作。如果你想将程序用在不同序列号的控制板，必须或者重新改变控制板的分配或者改变控制板的序列号。要绕过序列号的问题还有第二种途径：“**按照列表次序(sequence)**”。如果你选择这项，此时程序储存顺序号而不是序列号。尽管在 USB 总线上，添加或删除控制板可能会导致冲突，程序运行却能依旧在任何控制板上运行。

7.3 在子程序中控制板的分配

一般，在主程序属性窗口中可以把程序所有的控制板分配好。然而，你也可以在子程序属性窗口中输入控制板分配。这样，你在子程序中可以使用来自主程序或者子程序中的控制板分配。如果两个控制板分配同名，那么在子程序中的控制板分配名称拥有优先权。在这里举个例子，你可以定义 **EM1** 作为主程序中访问主控制板的名字，但是同时代表特定子程序中的扩展板。如果你想控制整个车间，并且每台机器由各自的控制板控制，这种应用无疑非常有价值。这样的话，你可以把为每台机器独立开发控制程序，且每个主程序都访问 **EM1**。然后你把所有的机器的主程序作为子程序安装在一个整体程序中。在整体程序中，你只需要改变控制板的分配，而不需要再改变各个独立输入和输出的名称了。

7.4 技巧和窍门

如果你想将一个原来为一块带三块扩展板的 ROBO 接口板开发的程序，运行到两个各带扩展接口板的智能接口板上。你可以使用如图所示的接口板配置。在这里，将扩展模块 2 和 3 替换为另一块接到在端口 COM2 上的带扩展板的智能接口板。

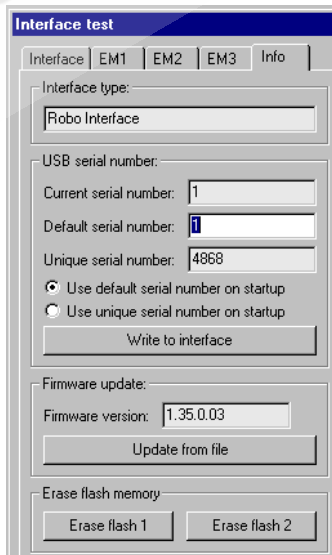


7.5 改变控制板序列号

在缺省状态下, 每块 ROBO 接口板和 ROBO I/O 扩展板都有一个相同的序列号。如果你只是想在计算机上使用一个控制板, 这是完全适应的。因为在此时, 所有的接口对于计算机来说都是相同的, 这时改变接口板是没有任何问题的。但是, 如果你想在计算机 USB 总线上运行多块控制板, 就必须预先更改接口板的序列号。这样一来, 计算机可以就能区分各个接口板, 并为它们分配地址。另一方面, 如果你要为接到串口的控制板分配地址, 就没有必要了。

改变控制板序列号的步骤如下:

- 把控制板**单独**连接到计算机的 USB 总线。
- 在工具栏上, 点击“**编程环境(Environment)**”切换到 ROBO 接口板的编程环境。
- 单击工具栏中的 **COM/USB** 按钮并选择“USB 端口(port)”项。
- 然后按下工具栏上的 **Test** 按钮, 打开接口测试窗口, 并选择 **Info** 选项卡。
- 在“**控制板类型(Interface type)**”一栏, 显示了控制板的类型, 例如: **ROBO Interface** 或是 **ROBO I/O Extension**。
- 在 **USB serial number** 一栏, 你可以设置在接口启动时所使用的序列号, 每个接口都有两个内置的序列号, 一个缺省的序列号(**default serial number**), 如果你没有改变设置, 那么就指定为 1; 另一个是**特定序列号(unique serial number)**, 这是所有的接口都不同并且不能再写入的。在 USB 总线上使用多块控制板的最简单的方法是, 选中“**使用特定序列号(Use Unique serial number)**”选项。这样, 每个控制板都确定的分配到一个正确的序列号。如果你在一个模型中使用多个控制板, 毫无疑问, 记住所有的序列号是不切实际的。所以, 在这种情况下,



你将缺省状态下的控制板序列号设置为，诸如 1, 2, 3 等等，可以简化此问题。在你重设和选择好序列号之后，你还必须单击按钮 **“Write to Interface”** 来写入控制板。改变完序列号之后，你必须将控制板断电并重新上电。

警告：如果改变了序列号，必须重新安装驱动程序，这要求有 Windows 的管理权限。如果你改变了序列号，但是不能重装驱动程序，这是由于你没有系统管理权，就不能通过 USB 接口来访问控制板了。这种情况下，你可以关闭接口板电源，并在重新上电的时候按住端口按钮，那么控制板就会在串口 1 启动，而且会再次由已安装的驱动程序识别。然而，这么做不能永久地恢复序列号，例如下一次启动时如果不按住端口按钮，上一次的序列号将会重新保存。要永久地恢复序列号，你必须进行如上所述的步骤。

- 最后，在 **“更新固化程序(Update firmware)”** 一栏，如果慧鱼公司提供了一个新版本的控制板固化程序，可以更新你的 ROBO 接口板的内部控制程序。

8 编程模块概览

ROBO PRO 软件的所有有效的编程模块排列如下，而且它们是根据模块窗口中的顺序进行描述的。

Cedutech®

8.1 基本模块（级别 1）

Cedutech®

8.1.1 开始

程序流程都是由“开始”模块作为开头的。如果程序不是由此模块开头，流程就无法执行。假如一个程序由几个流程组成，每一个流程必须由“开始”模块开头。各个不同的流程就同时开始启动。



“开始”模块属性一般无需更改。

Cedutech®

8.1.2 结束

如果一个流程结束，最后一个模块的出口应该连到“结束”模块。流程也可以在各个不同的地方用此模块终结，也可以将各个不同模块的出口连接到同一个“结束”模块。但是，也有可能流程是个没有结束的循环，不含“结束”模块。

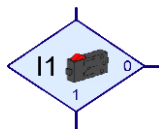


“结束”模块**没有**属性可以改变，所以，如果鼠标右键点击模块，则不像其它大多数模块会有属性窗口打开。

Cedutech®

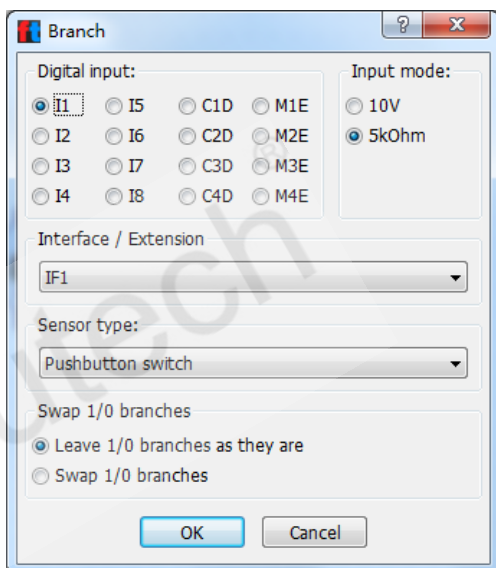
8.1.3 数字量分支

根据某一个数字量输入 **I1-I8** 的状态，在其中一个方向上你可以用此分支来控制程序进程。比如，数字量输入的某个传感器闭合(=1)，则程序分支走“**1**”出口。反之，如果输入断开(=0)，则程序分支走“**0**”出口。

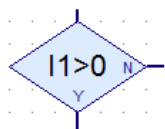


如果鼠标右键点击模块，会有属性窗口打开：

- 按钮 **I1** 到 **I8** 允许选择一个要查询的 ROBO TX 控制板的通用输入端。
- 按钮 **C1D-C4D** 允许将 ROBO TX 控制板的 C1-C4 输入端其中一个作为简单的数字量输入。
- 按钮 **M1E-M4E** 允许查询其中一个 ROBO Pro 内部输入。当使用“**扩展电机控制(Extended Motor Control)**”控制的电机达到预定位置，这些变量被置为 1。
- 在“**接口板 / 扩展板 (Interface/extension)**”一栏，可以选择使用主控制板或扩展板的输入端口。详细信息可以参见[第 7 章“控制多个控制板”](#)。
- 在“**传感器类型(Sensor type)**”一栏，可以选择连接到输入端的传感器。微动开关是最常用的数字量输入形式，但也经常用光电晶体管和干簧管。ROBO Pro 可以根据选择的传感器类型自动更改**输入模式(Input mode)**，在级别 4 以上级别，也可以单独修改输入类型。
- 在“**交换 1/0 分支位置(Swap Y/N branches)**”一栏，可以交换分支中“1”和“0”出口的位置。一般情况下，“1”出口在下部，“0”出口在右边。但有时候将“1”出口放到右边会更实用，按一下“**Swap Y/N branches**”，点击 **OK** 关闭对话框后，这两个出口就互换了。



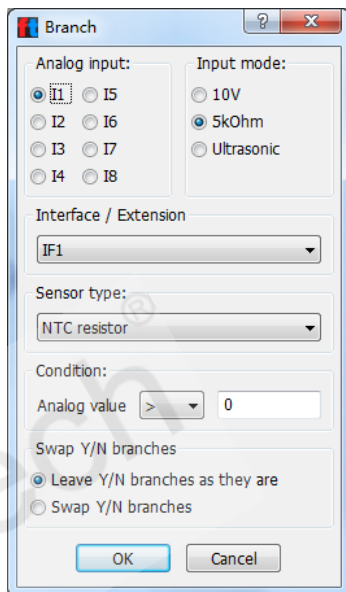
8.1.4 模拟量分支



除了数字量输入，TXT 和 TX 控制板的 I1-I8 端口还可以作为模拟量使用。用此模块可以将模拟量输入值和固定值进行比较，根据比较的结果，来确定分支的“**Yes(Y)**”或者“**No(N)**”出口。

如果鼠标右键点击模块，会有属性窗口打开：

- 在“**模拟量输入(Analog input)**”一栏，可以选择某一个要查询的 TXT 或 TX 控制板的通用输入端。
- 在“**控制板 / 扩展板 (Interface/extension)**”一栏，可以选择所需的是主控制板或扩展板的输入信号。详细信息可以参见[第 7 章“控制几个控制板”](#)。
- 在“**传感器类型(Sensor type)**”一栏，可以选择连接到输入端的传感器。ROBO Pro 可以根据选择的传感器类型自动更改**输入模式(Input mode)**，在级别 4 以上级别，也可以单独修改输入类型。

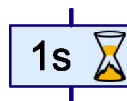


传感器	输入模式	显示值
NTC 热敏电阻，光敏电阻	模拟量 5kOhm	0-5000Ohm
颜色传感器	模拟量 10V	0-10000mV
超声波距离传感器(TXT 和 TX 控制板所用为 3 线传感器，货号为 133009)	距离	3-400cm

- 你可以在[8.7.1 章节“通用输入”](#)找到更多有关模拟量输入的信息。
- 在“**条件(condition)**”一栏，可以选择一个比较算法，比如小于 (<) 或者大于 (>)，并输入比较值。
- 在“**交换 Y/N 分支位置(Swap Y/N branches)**”一栏，可以交换分支中“Y”和“N”出口的位置。一般情况下，“Y”出口在下部，“N”出口在右边。但有时候将“Y”出口放到右边会更实用，按一下“**Swap 1/0 branches**”，点击 OK 关闭对话框时，这两个出口就互换了。

8.1.5 延时

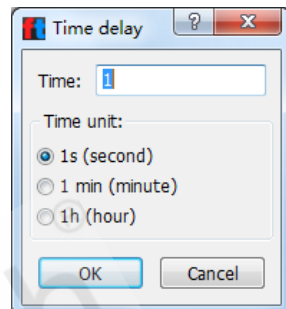
用**延时模块**可以使流程执行延迟一个你所设定的持续周期。



如果鼠标右键点击模块，会有属性窗口打开：你可以在窗口中输入所要延迟的时间（秒、分钟或者小时）。延时时间范围可以从 1 毫秒到 500 小时。然而，延时时间越长，精度越低。

下表显示了各个时间段延时的精度。

延迟	精度
至 30 秒	1/1000 秒
至 5 分钟	1/100 秒
至 50 分钟	1/10 秒
至 8.3 小时	1 秒
至 83 h 小时	10 秒
至 500 小时	1 分钟



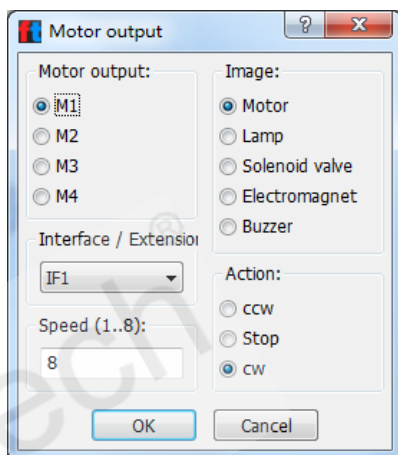
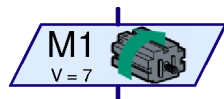
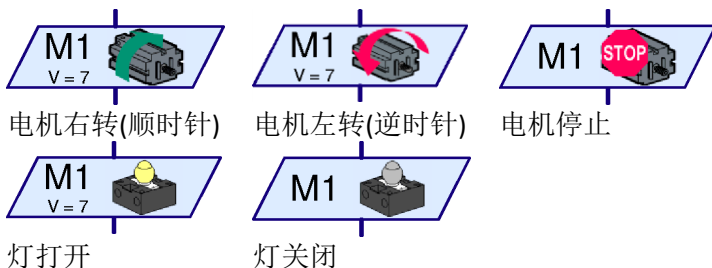
8.1.6 电机输出

用“电机输出”程序模块可以改变控制板的两极输出 M1-M4 中某一个的状态。控制板的输出可以是电机，也可以是灯或者电磁铁。对于电机，可以设置它的转向和速度。

如果鼠标右键点击模块，会有属性窗口打开：

- 在“电机输出(Motor output)”一栏，可以设定选用 M1 至 M4 中的一个作为输出。
- 在“控制板 / 扩展板 (Interface/extension)”一栏，可以选择所需的是主控制板或扩展板的输入信号。详细信息可以参见[第 7 章“控制几个控制板”](#)。
- 在“类型(Image)”一栏，可以选择代表连接到输出的慧鱼器件的图示。
- 在“动作状态(Action)”一栏，可以设置输出该如何动作。对于电机，可以设置电机左转（逆时针）或者右转（顺时针）或者停止。如果在电机输出上接了一个灯（见下面的灯输出提示），可以打开或者关闭它。
- 最后，可以在“1”和“8”之间指定一个速度(Speed)或强度(Intensity)。8 是最大速度，亮度或者磁场强度；1 最小。在停止或者关闭的情况下，通常不需要指定速度。

这里列举了一些动作符号和图示：



小贴士: 有时候电机只朝一个方向运行, 比如输送带电机。这种情况下, 可以把电机接到灯输出, 这样可以少用一个输出端口。

Cedutech®

8.1.7 带编码器电机（级别 1）

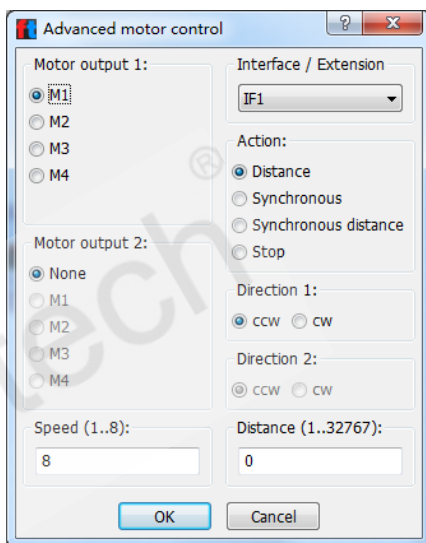
带编码器电机(Encoder Motor)程序模块在级别 1 就可以使用，允许带编码器电机的多项控制。

用这个模块，可以控制单个电机转动一定的脉冲数，或者控制两个电机同步转动，是否需要转动一定距离可以自选。如果鼠标右键点击模块，会有属性窗口打开：



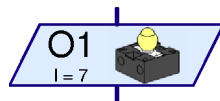
- 在“**动作状态(Action)**”一栏，可以设置控制电机转动一定脉冲**(Distance)**，即输出轴转动一定角度；两个电机同步转动**(Synchron)**；两个电机在同步转动情况下转动一定脉冲**(Synchron Distance)**，即输出轴转动一定角度。取消这些动作并停止电机，选择停止(stop)按钮。
- 在“**电机输出 1/2(Motor output 1/2)**”，你可以选择电机输出控制的端口，可以选择控制一个或两个电机。
- 在“**控制板 / 扩展板 (Interface/extension)**”一栏，可以选择所需的是主控制板或扩展板的输入信号。详细信息可以参见[第 7 章“控制几个控制板”](#)。
- 在“**方向 1/2(Direction 1/2)**”一栏，可以选择电机转向。
- 在“**速度(Speed)**”一栏，可以控制电机的转速，如果同步控制两个电机，两个电机速度相同。
- 最后，在“**距离(Distance)**”一栏，可以所需电机转动的脉冲数，即电机输出轴转动的角度。

在 [12.6.1 带编码器电机（级别 1）](#) 章节，可以获取更多关于使用这个模块的信息。



8.1.8 灯输出（级别 2）

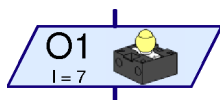
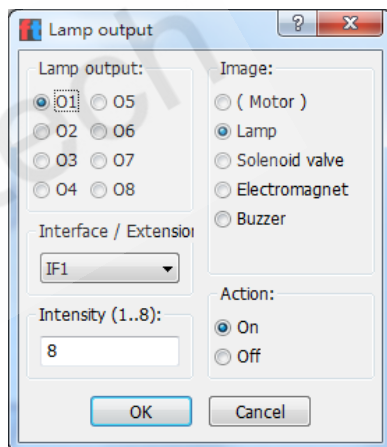
用**灯输出(Lamp output)**编程模块，可以开关接口板的任一个单极输出 O1-O8。接口板的输出既可以成对地用作电机输出（见前一节）也可以用作单个的灯输出 O1-O8。与电机输出不同，灯输出只占用一个接线端。这就是为什么可以控制 8 个灯或者电磁阀。可以将灯的另一个接线端连接到接口板的接地插孔(⊥)。



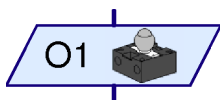
小贴士：如果你连接四组灯或者电机，也可以用电机输出来接灯。这样更实用，因为这样可以将灯的两个接线端直接连接到控制板的输出，而不是必须将所有负极都连到接地插孔。

如果鼠标右键点击模块，会有属性窗口打开：

- 在“**灯输出(Lamp output)**”一栏，你可以设置 O1-O8 中的任一个作为要使用输出。
- 在“**控制板 / 扩展板 (Interface/extension)**”一栏，可以选择所需的是主控制板或扩展板的输入信号。详细信息可以参见[第 7 章“控制几个控制板”](#)。
- 在“**类型(Image)**”一栏，可以选择代表连接到输出的慧鱼器件的图示。
- 在“**动作状态(Action)**”一栏，可以设置使输出如何动作。你可以打开或者关闭灯。
- 最后，你也可以指定 1-8 之间的一个**强度(Intensity)**。8 对应的亮度最大，1 最小。在灯关闭的情况下，自然就不需要再定义强度了。这里列出了灯的各种动作的图示符号。



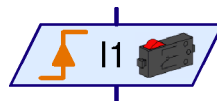
灯打开



灯关闭 f

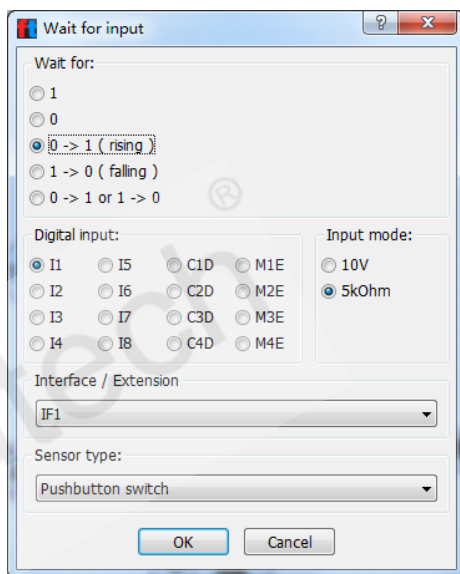
8.1.9 输入等待

“输入等待”模块的功能是，等待直到控制板的某个输入变为特定状态或者其状态由某一特定方式改变。

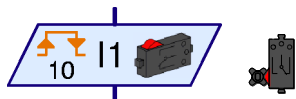


如果鼠标右键点击模块，会有属性窗口打开：

- 在“等待(Wait for)”一栏，可以选择信号变化的类型或者所等待的信号状态。如果你选择 **1** 或者 **0**，模块一直等待，直到输入信号闭合(1)或者打开(0)。如果你选择 **0 -> 1** 或者 **1 -> 0**，模块一直等待输入信号状态从打开到闭合变化(0->1) 或者从闭合到打开 (1->0)。最后一种情况是，模块一直等待，直到输入信号状态变化。而不管是从打开到闭合，反之亦然。为帮助你更好地理解，请见 [3.6.2“输入等待”章节](#)，如何用判断模块来模拟这个模块。
- 在“数字量输入(Digital input)”一栏，可以确定读取通用输入端 **I1 - I8** 的任一输入信号。其他端口的详细描述请参考 [8.1.3 “数字量分支” 章节](#)。
- 在“控制板/扩展板(Interface/extension)”一栏，可以选择所需的是主控制板或扩展板的输入信号。详细信息可以参见 [第 7 章 “控制几个控制板”](#)。
- 在“传感器类型(Sensor type)”一栏，可以选择连接到输入端的传感器。微动开关是最常用的数字量输入形式，但也经常用光电晶体管和干簧管。ROBO Pro 可以根据选择的传感器类型自动更改输入模式(Input mode)，在级别 4 以上级别，也可以单独修改输入类型。



8.1.10 脉冲计数器

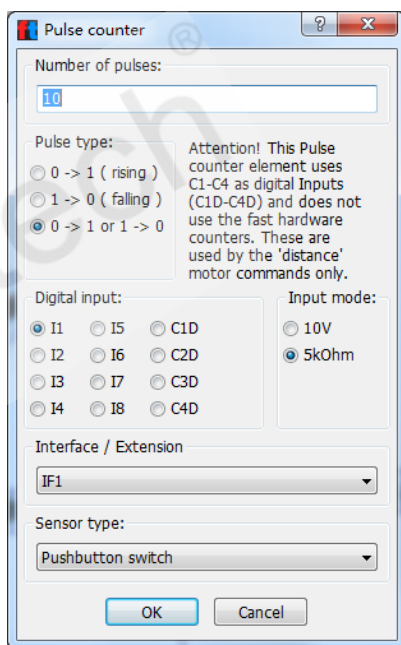


许多慧鱼模型都用到了脉冲齿轮，这些齿轮每一圈碰触传感器四次。使用这些脉冲齿轮，可以使电机运行精确定义的圈数，而不是一段给定的时间。为了实现此目的，必须对控制板输入端的脉冲进行计数。这样，由“脉冲计数器”模块来等待一个用户自定义的脉冲数。

注意：控制带编码器的电机有特殊的模块，使电机控制更加精确。详情请参考 [8.1.7 带编码器电机（级别 1）](#) 章节。

如果鼠标右键点击模块，会有属性窗口打开：

- 在“脉冲类型(Pulse type)”一栏，可以选择所要计数的脉冲类型。如果选择 0 -> 1（上升沿），模块一直等待输入的状态从打开变为闭合(0->1)，跳变的次数可以“脉冲数量(Number of pulses)”中定义。如果选择 1 -> 0（下降沿），模块一直等待输入的状态从打开变为闭合(1->0)，跳变的次数同样在“脉冲数量(Number of pulses)”中定义。然而，对于脉冲齿轮，第三种可能性更常用：模块对 0 -> 1 和 1 -> 0 的变化都进行计数，这样，脉冲齿轮每转一圈可计得 8 个脉冲。
- 在“数字量输入(Digital input)”一栏，可以选择读取控制板中 I1 - I8 的任一输入信号。C1D-C4D 是选择控制板的计数端口，然而这里无需使用这些端口适用于快速计数，除非最大频率达到 100Hz。
- 在“控制板/扩展板(Interface/extension)”一栏，可以选择所需的是主控制板或扩展板的输入信号。详细信息可以参见 [第 7 章“控制几个控制板”](#)。
- 在“传感器类型(Sensor type)”一栏，可以选择连接到输入端的传感器。微动开关是最常用的数字量输入形式，但也经常用光电晶体

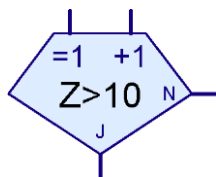


管和干簧管。ROBO Pro 可以根据选择的传感器类型自动更改**输入模式(Input mode)**，在级别 4 以上级别，也可以单独修改输入类型。

Cedutech®

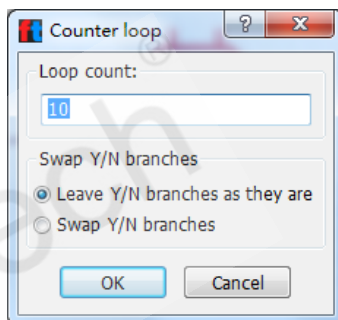
8.1.11 循环计数器

用“循环计数”模块，你可以很方便地让程序的某一部分执行多次。“循环计数”有一个内置计数器。如果循环计数从**=1**进入，计数器则置为1。如果循环计数从**+1**进入，则计数器加1。根据计数器的值是否大于你预设的值，循环计数来选择“**Yes(Y)**”或者“**No(N)**”为出口。你可以[3.6.4 “循环计数”](#)章节中找到一个范例。



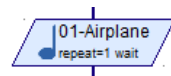
如果鼠标右键点击模块，会有属性窗口打开：

- 在“循环计数(Loop count)”一栏，可以输入在“**Yes(Y)**”出口激活之前，“循环计数”从“**No(N)**”出口执行的次数。输入值必需为正。
- 如果选中了“交换Y/N分支位置(Swap Y/N branches)”，你点击“OK”，关闭窗口时，**Y**和**N**连线就会交换。根据**Y**和**N**连线的位置，程序中被重复执行的部分将会在“循环计数”模块的右部或者下部。



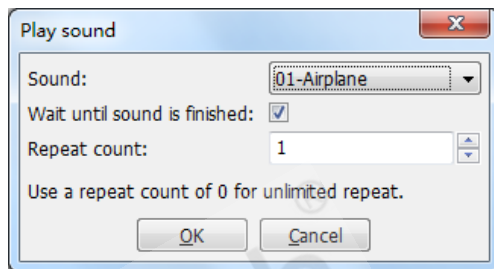
8.1.12 音乐播放器

ROBOTICS TXT 控制板包含内置喇叭，可以发出各种音乐响声。



音乐播放器模块的属性窗口：

- 在“声音(Sound)”一栏，可以选择所要播放的音乐，内置 29 种音乐。
- 选中“等待音乐结束 (Wait until sound is finished)”，则该程序进程等待直到音乐结束才进入下一个模块。
- 在“重复次数 (Repeat count)”一栏，输入音乐重复的次数。若在输入为 0，则无线循环。



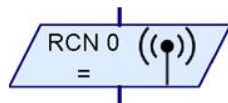
8.2 发射接收模块（级别 2-3）

在这些模块中，你将了解关于通过蓝牙发射与接收信息的模块。

Cedutech®

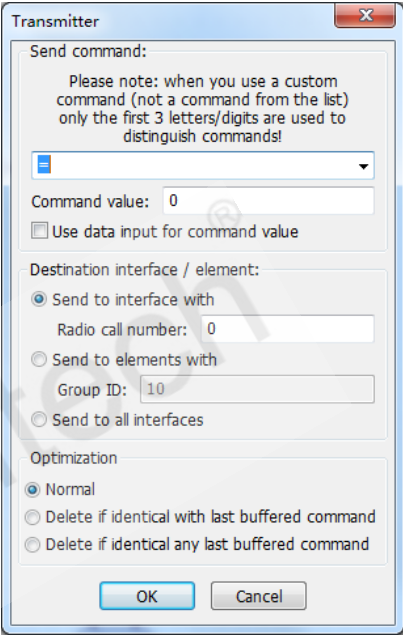
8.2.1 发射器（级别 2）

使用信号发射器，可以通过蓝牙发送指令或信息。这样，几个机器人之间就可以进行交流。



发射器模块的属性窗口

- 在“发送指令(Send command)”栏，你可以输入要发送的指令，从级别 3 开始，还可以附带指令的数值。每条指令都包括名称和数值，数值可以通过数据输入模块输入。对于指令名称，如果未被包含在列表中，则只有头 3 个字母或数字能够被识别。虽然你可以输入更多字母，但是，例如“Hello”，“Help”，“helicopter”都表示同一条信息，因为每个名称都以“Hel”开头。另外大小写不区分，特殊符号（空格,!?,%）都认为相同，例如“XY!”与“XY?”也表示同一条信息。数字当然能够区分，例如“XY1”与“XY2”就表示不同的信息。

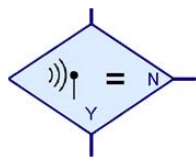

- 在“目标控制板/模块(Destination interface/element)”一栏，你可以选择指令发送到哪一个控制板。通常情况下，你要通过**无线通信号码(radio call number)**向一个特定的控制板发送信息，也可以无需设置无线通信号码，向所有附近控制板发送信息。从级别 4 开始，可以向特定的控制板**组群(group)**发送信息，组群的接收地址可以是 10 到 255 的数字。你可以改变组群的地址，以区分信息要发送给哪些控制板。组群的地址从 10 开始，因为 0 到 9 被保留。通过接收器模块，你可以了解到更多关于被保留数字的信息。
- 在“优化(optimization)”（从级别 4 开始）一栏，你可以调整同样的指令是否要被多次发送。对于许多指令，发送一次与一次接一次的发送并无太大区别。但是没有优化，指令会被多次发送直到占满信息交换的缓存区，使得其他指令不能被快速发送。因此，需要将

同样的指令删除。通常，你希望**删除与上一次缓存相同的指令(delete if identical to last buffered command)**。如果在这种模式下发送信息，例如，2 个 Start 和 2 个 Stop 指令，只有 1 个 Start 和 1 个 Stop 指令会被发送，如果你发送 Start,Stop,Start,Stop，这样一个接一个快速发送，而不是一次发送两个相同的指令，指令发送就和预期一样，不会改变。然而，你还可以这样删除要指令，如果其**与缓存中任一指令相同(identical to any buffered command)**。相反，对于许多指令，优化显得并不合理，选择正常(normal)就可以了。例如发送“**添加(Add)**”指令，这会在向列表(list)添加数据时使用。对于列表，添加一次和添加两次就不同了。在级别 2 时，默认为“**删除与上一次缓存相同的指令(delete if identical to last buffered command)**”。

Cedutech®

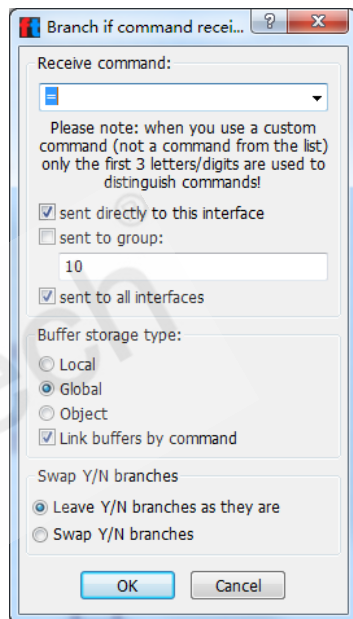
8.2.2 接收器（命令接收分支，级别 2）

接收器指令与发射器指令配合使用，基于接收或未接收到指令，分支会有 Y(YES)或 N(No)端口。



接收器模块的属性窗口

- 在“**接收指令(Receive command)**”栏，你可以输入要接收的指令，与发射器模块相同，只有头 3 个字母或数字能够被识别。然后你需要选择是只接收发射到这个控制板的指令，通过无线通信号码实现，还是接收对所有控制板的指令，两者可以同时选择。如前面所述，从级别 4 开始，你可以向特定的控制板组群发送信息，这些信息可以被选定的控制板组群内所有的控制板所接收。组群地址可以设定为 10 到 255 的数字，组群地址 0 到 8 则对应无线通信号码 0 到 8，组群地址 9 代表发送信息到所有控制板。当发送信息时，发送到组群 1 与发送到无线通信号码为 1 的控制板没有区别。但是对于接收模块，不能在这里设置，因为每个控制板有自己的无线通信号码。但是注明接收组群 1 到 8 的信息，则可以接收信息，也就是除了本身无线通信号码为特定的控制板外，多了一个接收信息的控制板。但是对于将组群地址设为小于 10 的值，必须在级别 5 中才能实现。
- 在“**缓存存储类型(Buffer storage type)**”一栏，可以选择接收的指令是作为**全局变量**或**局部变量存储**。如果选择全局变量，即使子程序未运行，也会接收指令。
- 在“**目标控制板/模块(Destination interface/element)**”一栏，你可以选择指令发送到哪一个控制板。通常情况下，你要通过**无线通信号码(radio call number)**向一个特定的控制板发送信息，也可以无需设置无线通信号码，向所有附近控制板发送信息。从级别 4 开始，可以向特定的控制板**组群(group)**发送信息，组群的接收地址可以是 10 到 255 的数字。你可以改变组群的地址，以区分信息要发送给哪



些控制板。组群的地址从 10 开始，因为 0 到 9 被保留。通过接收器模块，你可以了解到更多关于被保留数字的信息。

- 在“**交换 Y/N 分支位置(Swap Y/N branches)**”一栏，可以交换分支中“Y”和“N”出口的位置。一般情况下，“Y”出口在下部，“N”出口在右边。但有时候将“Y”出口放到右边会更实用，按一下“**Swap 1/0 branches**”， 点击 **OK** 关闭对话框时，这两个出口就互换了。

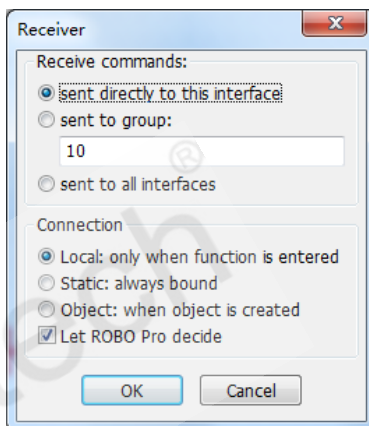
Cedutech®

8.2.3 接收器（级别 3）

上一节介绍的接收器模块主要在级别 2 中使用，可以接收指令，但是不能接收指令的数值。级别 3 下的接收器指令，正好相反，只能接收指令的数值。你无需注明该模块需要接收哪一个命令，这个模块将所有接收到的指令数值输出。

接收器模块的属性窗口

- 在“接收指令(Receive command)”栏，你可以输入要接收的指令，与发射器模块相同，只有头 3 个字母或数字能够被识别。然后你需要选择是只接收发射到这个控制板的指令，通过无线通信号码实现，还是接收对所有控制板的指令，两者可以同时选择。如前面所述，从级别 4 开始，你可以设定特定的组群。你可以在 [8.2.1 发射器（级别 2）](#) 和 [8.2.2 接收器（命令接收分支，级别 2）](#) 中了解到更多关于组群接收的内容。在级别 2 和级别 3 中，你只能选择接收所有信息或接收发送到这个控制板的信息。但是你想接收多个不同目的地的信息，可以将多个接收器模块设置为接收不同区域，特别地，不同目的地区域可以是不同的组群地址。

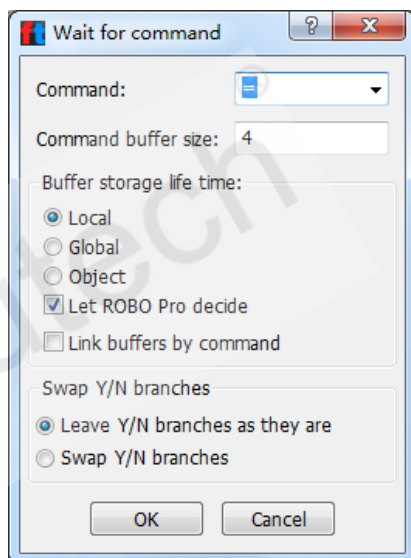


8.2.4 等待命令（级别 4）

“等待命令(Wait for Command)”的使用方法与[接收器（命令接收分支，级别 2）](#)类似，用于等待指令。然而这个模块并不是等待发送到控制板的指令，而是等待左侧输入端口的指令。如果将接收器（级别 3）模块与其连接，就形成了一个级别 2 下的接收器模块。同时，这个组合还有数据输出的功能，在其右侧的 V 端口。该模块在接收到特定指令后，程序流线从 Y 端口出，指令数据从右侧 V 端口输出。你可以在[6.1 进程命令](#)章节找到示例。

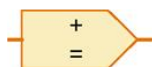
等待命令模块的属性窗口

- 在“指令(Command)”栏，你可以输入要等待的指令，你也可以输入自定义的指令，当然只能识别头 3 个字母或数字。详细内容请查看[8.2.1 发射器（级别 2）](#)章节。
- “指令缓存容量(Command buffer size)”只在级别 5 才能设置。作为级别 2 下的接收器，这个模块能够设置接收多少个指令。由于等待指令同时要注意指令的数值，最大缓存容量有限。一般情况，接收 4 个指令的缓存容量就可以了，因为程序指令在大多数情况下都是立刻接收到的。



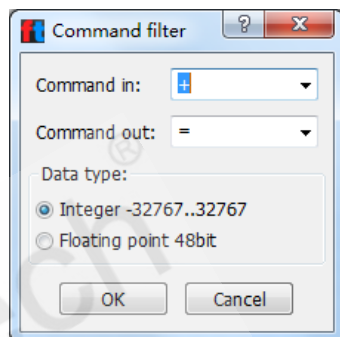
8.2.5 命令过滤器（级别 4）

使用“命令过滤器(Command Filter)”，可以重新定义指令名称。当一个特定的指令被发送到该模块的左侧入口端，模块发送另一个命令到与右侧出口端相连的模块，同时保持指令数值不变。所以，你可以讲一个电机指令转换为一个赋值“=”指令，具体示例可以参考 [6.2 命令过滤器](#) 章节。



命令过滤器模块的属性窗口

- 在命令过滤器模块中有个指令可供设置：**输入指令(Command in)**和**输出指令(Command out)**，你也可以输入自定义的指令，当然只能识别头 3 个字母或数字。详细内容请查看 [8.2.1 发射器（级别 2）](#) 章节。
- 在“数据类型(Data type)”你可以选择发送或接收指令中携带的数据类型，可以是整数或浮点数。具体请参考 [13 章小数功能](#)。



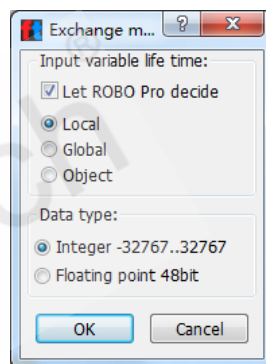
8.2.6 更换指令数据（级别 4）

与命令过滤器更换指令名称（见前一节）相似，**更换指令数据(Exchange Message)**模块更换指令的数据。与命令过滤器配合使用，可以将一个指令变为多个不同名称，不同数值的指令。例如，如果你要控制一个小车，它需要理解向左，向右或前进之类的命令，你可以将向左“left”指令通过命令过滤器转换为赋值“=”指令，另外通过更换指令数据模块，还可以将赋值指令中的数据变为 0 或负数，然后发送给另一个电机。需要改变数值的指令连接到 **C** 输入端，新的数据连接到 **V** 输入端。



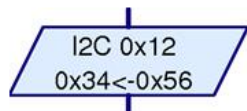
更换指令数据模块的属性窗口

- 在“**输入变量适用范围(Input variable life time)**”一栏，你可以选择 **V** 端数据类型，是全局变量还是局部变量。一般选择“Let ROBO Pro decide”。
- 在“**数据类型(Data type)**”你可以选择发送或接收指令中携带的数据类型，可以是整数或浮点数。具体请参考 [13 章小数功能](#)。



8.2.7 I2C 写入（级别 4）

I2C 写入(I2C Write)模块将指令或数据发送到 TXT 或 TX 控制板的 I2C 接口。对于 TXT 控制板，I2C 端口在 EXT 端口，对于 TX 控制板，I2C 端口在 EXT2 端口。标准 I2C 接口可以将其他厂商的传感器和执行器连接到 TXT 或 TX 控制板。使用 I2C 接口需要丰富的电气元件使用经验和相应的测量仪器。

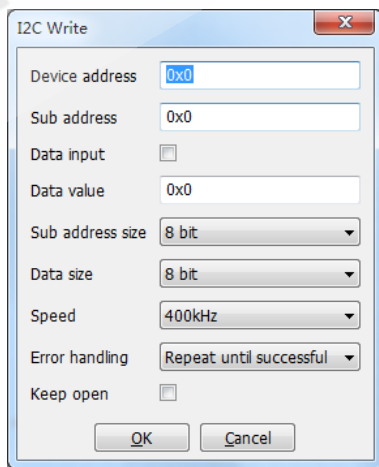


许多经常使用的 I2C 设备可以在 **Library\I2C** 列表下找到，你可以用 ROBO Pro 软件直接打开库文件，库主程序包含一个测试程序，库文件在 ROBO Pro 安装文件夹下的 Library\I2C。

I2C 写入模块通过 I2C 端口发送 1 到 4 字节的地址字节。首先，发送 7 位设备地址，然后是写入位，之后发送一个 0-2 字节的长子地址，最后发送 1 或 2 个字节的数据。I2C 协议中的一点优势是子地址和数据没有什么不同，然而许多 I2C 设备都需要在发送设备地址后，先发送子地址，后发送数据。精确的协议可以在 I2C 设备的参数表中找到。

I2C 写入模块的属性窗口

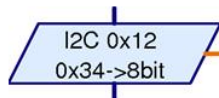
- 在“**设备地址(Device address)**”写入 7 位设备地址（不包含写入/读取位），特别地，有些设备的地址为 8 位（不包含写入/读取位）。这种情况下，有必要将地址除 2，例如用 0x60 代替 0xC0。
设备地址 0x50...0x57(=0xA0...0xAF) 只是针对使用 TX 控制板的情况，对于用于其他设备。
- 在“**子地址(Sub address)**”中写入 8 位或 16 位的子地址，请参考下面**子地址容量(Sub address size)**。
- 在“**输入数据(Data input)**”一栏，可以选择是在下方**数据值(Data value)**中直接写入数据，或从外部获得。
- 在“**数据值(Data value)**”中写入所需数据，当然这是没有使用外部数据的情况下。



- 在“子地址容量(Sub address size)”一栏，你可以选择是否使用子地址。并非所有的 I2C 设备都使用子地址，所以可能会选择无(none)。如果选则 16 位子地址，你需要选择怎样传输，是从最高位(MSB:Most Significant Byte)开始，或者从最低位(LSB:Least Significant Byte)开始。
- 在“数据容量(Data size)”一栏，你可以选择发送 8 位或 16 位数据到 I2C 设备。对于 16 位数据，你可以选从最高位开始传输或从最低位开始传输，这与子地址的设置相同。
- 在“频率(Speed)”一栏，你可以选择 I2C 的时钟频率，可以是 100kHz 或 400kHz，当所有连接的 I2C 设备都支持 400kHz 时钟频率时，你应该选择 400kHz，否则就选择 100kHz。
- 在“错误处理(Error handling)”一栏，你可以选择如何处理发生控制板与 I2C 设备无法正常通信的情况。你可以选择“重复执行直至发送成功(Repeat until successful)”，“重复执行 10 次(Repeat 10 times)”或“立刻终止(abort immediately)”。对于后两个选项，在模块下端右侧会提供一个额外的错误端口。
- 当选中“持续打开(keep open)”，这个模块不会在最后向 I2C 总线发送结束指令。这允许使用其他 I2C 写入模块或 I2C 读取模块来写入更多数据或读取数据。如果读取和写入操作没有交替完成，或是在执行带有子地址的读取指令时，接下来 I2C 模块就不会再次发送设备地址。在 I2C 总线上，新的开始是读取和写入操作交替时，并不是一个结束开始的顺序。I2C 总线保持当前进程状态，除非 I2C 模块执行当前进程时没有勾选“keep open”，其他进程在此时使用 I2C 模块会无效。

8.2.8 I2C 读取（级别 4）

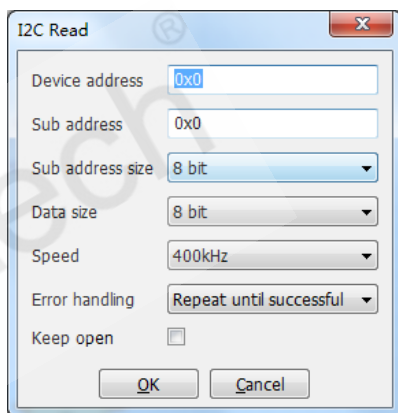
I2C 读取(I2C Read)模块读取 TXT 或 TX 控制板 I2C 接口的数据。描述与 I2C 写入模块相同。



当使用子地址时，I2C 读取模块首先在写入模式下发送地址字节，然后发送 1-2 字节子地址。然后模块在 I2C 总线上执行新开始，再次发送设备地址，这次在读取模式下，然后读取 1-2 自己的数据。如果子地址没有使用，会直接在读取模式下发送地址字节，然后读取数据。

I2C 读取模块的属性窗口

- 在“**设备地址(Device address)**”写入 7 位设备地址（不包含写入/读取位），特别地，有些设备的地址为 8 位（不包含写入/读取位）。这种情况下，有必要将地址除 2，例如用 0x60 代替 0xC0。
设备地址 0x50...0x57(=0xA0...0xAF) 只是针对使用 TX 控制板的情况，对于用于其他设备。
- 在“**子地址(Sub address)**”中写入 8 位或 16 位的子地址，请参考下面**子地址容量(Sub address size)**。
- 在“**子地址容量(Sub address size)**”一栏，你可以选择是否使用子地址。并非所有的 I2C 设备都使用子地址，所以可能会选择无(none)。如果选则 16 位子地址，你需要选择怎样传输，是从最高位(MSB:Most Significant Byte)开始，或者从最低位(LSB:Least Significant Byte)开始。
- 在“**数据容量(Data size)**”一栏，你可以选择发送 8 位或 16 位数据到 I2C 设备。对于 16 位数据，你可以选从最高位开始传输或从最低位开始传输，这与子地址的设置相同。
- 在“**频率(Speed)**”一栏，你可以选择 I2C 的时钟频率，可以是 100kHz 或 400kHz，当所有连接的 I2C 设备都支持 400kHz 时钟频率时，你应该选择 400kHz，否则就选择 100kHz。
- 在“**错误处理(Error handling)**”一栏，你可以选择如何处理发生控制板与 I2C 设备无法正常通信的情况。你可以选择“**重复执行直至**



发送成功(Repeat until successful)”，“重复执行 10 次(Repeat 10 times)”或“立刻终止(abort immediately)”。对于后两个选项，在模块下端右侧会提供一个额外的错误端口。

- “持续打开(keep open)”选项与 I2C 写入模块作用相同。

Cedutech®

8.3 子程序 I/O （级别 2-3）

在这个模块组中只有子程序所需要的程序模块。

Cedutech®

8.3.1 子程序入口（级别 2）



一个子程序可以有一个或多个子程序入口。主程序或者上层子程序通过这些入口将控制转入子程序。在插入上一级程序的子程序绿色符号中，每个子程序入口的一个连接端子都插在上边。这个符号上的连接线有一个相同的顺序（由上至下），正如子程序的功能设计中的子程序入口。如果鼠标右键点击模块，会有属性窗口打开。在窗口中可以给入口取个名字，然后此名字会在符号中显示。有关子程序更多的相关信息，可以参见[第 4 章“运行子程序”](#)。

Cedutech®

8.3.2 子程序出口（级别 2）



一个子程序可以有一个或多个子程序出口。子程序通过这些出口将控制转回主程序或者上层子程序。在插入上一级程序的子程序绿色符号中，每个子程序入口的一个连接端子都插在下边。这个符号上的连接线有一个相同的顺序（由上至下），正如子程序的功能设计中的子程序入口。如果鼠标右键点击模块，会有属性窗口打开。在窗口中可以给入口取个名字，然后此名字会在符号中显示。有关子程序更多的相关信息，可以参见[第 4 章“用子程序控制”](#)。

Cedutech®

8.3.3 子程序指令输入（级别 3）



通过子程序指令输入(Subprogram command input)模块，子程序可以关联到输入模块，比如主程序中，或者上一层的子程序的开关，或者来自变量模块的值，比如坐标。

在插入上一级程序的子程序绿色符号中，每个子程序指令输入的一个连接线都插在左边，这个符号上的连接线有一个相同的顺序（由左至右），正如子程序的功能设计中的子程序指令输入。如果鼠标右键点击模块，会有属性窗口打开。在窗口中可以给指令输入取个名字，然后此名字会在符号中显示。有关此模块使用更多的相关信息，可以参见[第 5.5 节“子程序的指令输入”](#)。

属性窗口

- 在“名称(Name)”中输入指令输入该模块的名称，在绿色子程序符号中，只显示头两个字母。
- 在“数据类型(Data type)”你可以选择发送或接收指令中携带的数据类型，可以是整数或浮点数。具体请参考 [13 章小数功能](#)。

Subprogram data input

Name:

Connection

☒ Let ROBO Pro decide

☒ Local: only when function is entered

☐ Static: always bound

☐ Object: when object is created

Data type:

☒ Integer -32767..32767

☐ Floating point 48bit

Passing mechanism

☐ Let ROBO Pro decide

☒ Command '=' only: the caller sends '=' commands

☐ Any command: the caller may send any command

OK Cancel

- 在“传递机制(Passing mechanism)”（级别 4 以上）一栏，你可以选择是否只接收赋值指令(Command '=' only)，或任意指令(Any command)。如果变量或接口端连接到子程序的查询端，应该选择只接收赋值指令。这种情况下，子程序输入端口储存最近发送的数值，当子程序被调用后立刻传递有效的正确数值。如果选择任意指令，你可以发送例如 **stop** 或自定义的指令。这些指令仅当使用子程序时才发送到子程序。这在控制一个包含电机模块的子程序时会显得有意义，例如，你希望在外部的电机发出指令。在 [6.3 “向子程序发送自定义指令” 章节](#) 中发现更多内容。

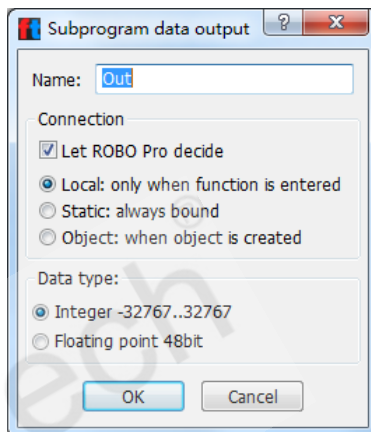
8.3.4 子程序指令输出（级别 3）



通过此模块，各种指令，比如向左、向右、停止等可以传送到主程序或者上层子程序的电机输出或者其他输出模块。在插入上一级程序的子程序绿色符号中，每个子程序指令输入的一个连接线都插在右边。这个符号上的连接线的顺序（由上至下），正如子程序的功能设计中的子程序指令输入。如果鼠标右键点击模块，会有属性窗口打开。在窗口中可以给指令输出取个名字，然后此名字会在符号中显示。有关此模块使用更多的相关信息，可以参见[第 5.5 节“子程序的指令输入”](#)。

属性窗口

- 在“名称(Name)”中输入指令输入该模块的名称，在绿色子程序符号中，只显示头两个字母。
- 在“数据类型(Data type)”你可以选择发送或接收指令中携带的数据类型，可以是整数或浮点数。具体请参考[13 章小数功能](#)。



8.4 变量，列表 ...（级别 3）

这一组的程序模块可以储存一个或多个数字值。使你可以开发带存储的程序。

Cedutech®

8.4.1 变量（全局）



Var
0

每个变量可以存储一个-32767 到 32767 之间的数值。变量的值可由连接一个赋值“=”模块到指令模块的左边来设定。（见[第 8.5.1 节 “=（赋值）”](#)）。在属性窗口中，也可以赋予变量一个初始值，并保持直到其收到第一个指令改变值。

ROBO Pro 只能为所有的变量模块建立一个同名而且变量类型为全局的变量。所有的同名全局变量都是一样的，而且有相同的值，即使它们出现在不同的子程序中。当其中一个变量模块通过指令改变了，所有其它的同名变量也被改变了。这一条对与局部变量并不适用。

除了“=”指令，变量也可以接受“+”和“-”指令。所以，比如说变量接到了一个指令“+ 5”，就将 5 加到了当前值上。对与“-”指令，指令传送的值就由当前值减去。

注意：

如果执行“+”或者“-”指令后，变量的值超出了所允许的范围，变量值就会加上或者减去 65536，以使得变量值回到有效范围中。由于此举通常并不受欢迎，你应该确信它不会发生为好。

每次变量值改变，它会传送一个带新值的“=”指令到所有与变量的指令输出相连的模块。如果你要监控变量的值，可以连接一个面板显示到变量的输出（见[第 8.7.7 节 “面板输出”](#)）。

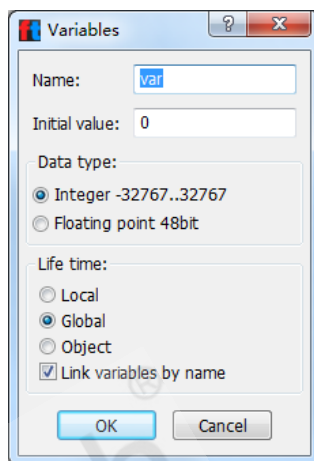
这里有关变量模块可以处理的所有指令的概要。

指令	值	作用
=	-32767 to 32767	将变量值设置为通过指令传递的值
+	-32767 to 32767	通过指令传递的值加到变量的当前值
-	-32767 to 32767	从变量的当前值减去通过指令传递的值

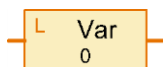
顺便提及，范围-32767 到 32767 是由计算机的二进制系统计算得来的，而不是通常的十进制系统。在二进制系统中，32767 是一个完全值，有点象十进制系统中的 9999。但是我们不必为此担忧，因为计算机会将所有的数都从二进制转换为十进制。我们只需要注意最大的数和在计算中的可能发生的溢出。

变量的属性窗口.

- 在“名称(Name)”一栏, 可以输入变量名。
- 在“初始值(Initial value)”一栏, 可以输入变量的初始值。变量保持这个值, 直到通过=, +, 或者-指令得到一个新的值。
- 在“数据类型(Data type)”你可以选择发送或接收指令中携带的数据类型, 可以是整数或浮点数。具体请参考 [13章小数功能](#)。
- “适用范围(life time)”选项, 只对子程序中变量有用, 更多详细信息将在下节讲述。



8.4.2 局部变量



所有同名的**全局**变量模块使用同一个变量，而且总是有相同的值。那大概是你所期望的，且通常是实用的。但是如果你在子程序中使用变量，会导致大问题。如果你的程序含有多个并行流程，在同一时间在多个场合会同时执行同一个子程序。在这种情况下，如果程序的所有流程中都使用相同的变量，会导致混乱。正因为如此，出现了**局部变量**。局部变量的运用和全局变量几乎相同，只有一点区别：局部变量只是在它被定义的程序中有效。即使在不同的子程序中两个局部变量同名，它们也是截然不同的独立的两个变量。即使同一个程序同时并行执行几个流程，每个流程中的子程序都有一套独立的局部变量。局部变量只在定义它们的子程序中发挥作用，所以在程序开始时局部变量并不被赋予初始值，而是在每次启动相关的子程序之时。因为子程序被调用多次，每次都完成相同的任务，所以在每次调用时变量都设置为相同的初始值，会实用得多。这样看来，局部变量不调用同一子程序的先前的存储区。

在主程序中，局部和全局变量作用方式相同，因为全局程序和主程序是同时启动的。然而，局部变量在程序执行时稍微有效些。另一方面，列表模块应该定义为全局的，因为全局变量的存储区比局部变量变更大。

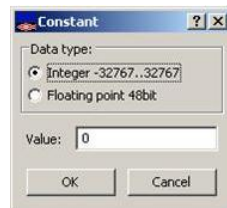
8.4.3 常量

0

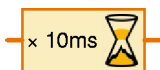
和变量一样，常量也有一个值，但是常量的值不能由程序来改变。如果子程序中总是使用一个相同的值，可以将一个常量和一个子程序符号的数据输入相关联。常量在运算符计算中也是非常实用的。在 [5.7 “运算符”章节](#) 的最后可以找到相关的例子。

常量的属性窗口。

- 在“数据类型(Data type)”你可以选择发送或接收指令中携带的数据类型，可以是整数或浮点数。具体请参考 [13 章小数功能](#)。
- “数值(Value)”一栏，输入具体常量的数值。

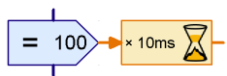


8.4.4 定时器变量



定时器变量的运用根本上和变量相似，甚至正常变量与静态变量的区别对于定时器也适用。唯一的区别在于定时器变量由存储的固定的时间间隔向下计时，直到其值为零。定时器的值一旦到达零，就保持在零。如果定时器的值变为负的，比如通过减法指令，其值会在下一个时间节拍回到零。

定时器变量向下计时的速度可以在其属性窗口中设定，在每节拍 1/1000 秒和每节拍 1 分钟之间。在操作的时候，你应该观察到定时器的精度取决于时间节拍的设置。比如，你如果设置一个时间段 1×10 秒，下一个时间节拍会在一个短时间后发生（比如说一秒钟），或者直到 10 秒钟之后。所以定时器只和时间节拍的设置精确度一致。所以，你宁可选择小的时间节拍，比如 10×1 秒或者 100×0.1 秒，而不是 1×10 秒。你应该只有在程序需要等待至少 1 小时的时候，才可以选择 1 分钟的时间节拍。那时，一分钟的误差不会造成大的差别。



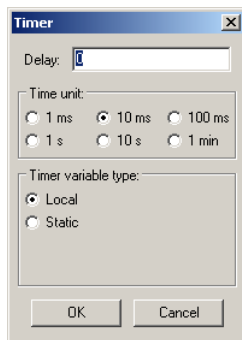
向下计时的节拍数通常是由赋值“=”指令来分配给定时器。在图示的例子中，要向下计时 100 个 10ms 的节拍。这对应于 $1000\text{ms}=1\text{s}$ 的一个时间间隔，其精度为 10ms。

定时器变量可以使你方便地解决困难的时间测量和延时问题。比如说，如果一个机器人打算 20 秒后停止搜索，你可以在开始搜索时设置一个定时器变量为 $20 \times 1\text{s}$ （或者 $200 \times 0.1\text{s}$ ），然后在搜索程序中有规律地查询定时器的值是否大于零。你也可以在搜索取得局部成功的地方将定时器复位带其初始值。

如果你要测量一个时间值，可以开始时将定时器变量设置为尽可能大的正值（30000 或者 32767）。这样一来，在定时器到达零前还剩下很多时间。如果你要知道到那时已经过了多少时间，可以将初始值减去当前的定时器的值。

定时器变量的属性窗口。

- 在“延时(Delay)”一栏，可以确定定时器变量的初始值。作为惯例，可以在这里输入 0，并在适当的时间用“=”指令来给定时器变量设定一个值。但是如果定时器假设是在程序或者子



程序开始时就开始运行，那么其相应值可以在这里输入。

- 在“**时间单位(Time unit)**”一栏，可以设定定时器变量在向下计时的时候，所用时间节拍的单位大小。
- 在“**定时器变量类型(Timer variable type)**”一栏，可以设定定时器变量为全局变量或静态变量。

Cedutech®

8.4.5 列表



列表模块相当于一个变量，其存储不止一个而是多个数值。变量中可以存储的数值的最大个数可以在其属性窗口中设定。

你可以在列表的末尾添加值或者在列表的末尾移除值。同时也可以改变或者读取列表的任何数值，也可以将列表中的任何数值和列表中的第一个数值交换。在列表中的中间或者开头无法直接插入一个数值。但是，你可以写一个子程序来实现这些功能。

下列的列表功能可以用传递指令到 **S**（写）输入来运用。下列指令可以传递到 **S** 输入：

指令	数值	作用
添加 	-32767 to 32767	添加指令可以将新的数值加到列表的末尾。整个列表就多了一个元素。如果列表已经达到了最大的数量，则忽略此指令。
删除 	0 to 32767	删除从列表末尾数指定位置的元素，与指令一起传递的数值是所要删除的序号。如果此号码大于列表中元素的数量，则所有的元素会被删除。如果号码为 0 或者为负，则忽略此指令。
交换 	0 to 32767	将指定的元素和列表的第一个元素交换位置，与指令一起传递的数值是所要交换的元素的位置号。

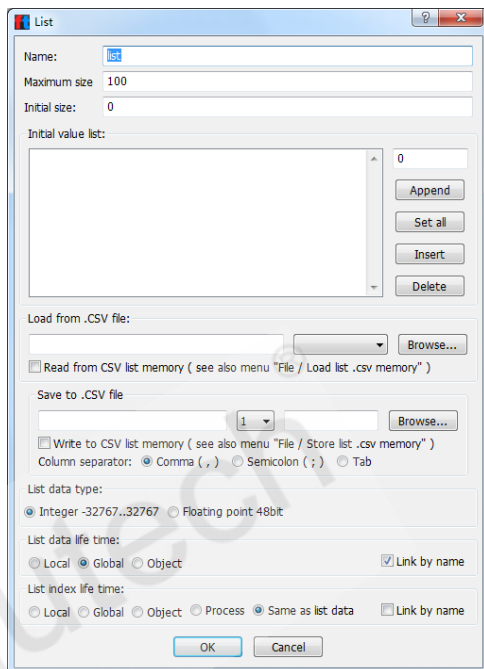
通过“**I**”（索引）输入，可以选择列表的一个特定的模块。具体来说，可以将所要的元素序号用“**=**”指令传递给“**I**”输入。第一个元素的序号为 0。可以用传递一个带所要的值的“**=**”指令，通过“**I**”输入和“**S**”输入，将另一个数值分配给所选定序号的元素。

通过“**I**”输入选定的元素可以通过“**R**”（读出）输出来查询，如果“**I**”输入，或者由“**I**”输入选择的模块的值改变了，列表将所选择序号的当前值传递到了那些连接到“**R**”输出的模块。

通过“**I**”输出，可以查询索引定义的“**I**”输入是否有效。如果 **N** 是列表元素的序号，0 和 **N-1** 之间数值一定会在“**I**”输入出现。如果是这样，“**I**”输出传递一个带值 **N** 的赋值“**=**”指令到所有已连接的模块。如果 **N-1** 大于列表元素的数量，那么“**I**”输出 0。

列表属性窗口

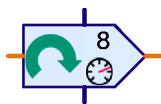
- 在“名称(Name)”一栏输入列表的名称。
- 在“最大数量(Maximum size)”一栏，可以输入列表的最大容量。这个容量无法用添加(Append)指令超出。
- 在“初始数量(Initial size)”一栏，输入开始时列表元素的数量。
- 在“初始值列表(List of initial values)”区域，可以输入列表元素的初始值。你可以用右边的按钮对序列进行编辑。
- 在“从.CSV 文件调用(Load from .CSV file)”一栏，可以选择一个 Excel 兼容的 .CSV 文件，列表可以从此文件中提取数值。在所选择区域的中部，可以选择.CSV 文件的哪一列数值用于此目的，文件直接加载并且显示在 List of initial values 下。当你开始执行程序或者执行下载操作，ROBO Pro 会多次尝试从文件加载当前值。如果不成功，则使用存储在 List of initial values 下的数值。
- 在“存入.CSV 文件(Save to .CSV file)”一栏，你可以指定一个文件，用来存储程序结束后列表的内容。然而，这项功能只对在线模式和静态列表（见下一点）有效。列表的内容写在文件的指定列。在“列分隔符(Column separator)”一栏，可以选择列表的单列是否应该用逗号或者分号分开。在使用 0.5 的国家，通常将逗号用作列分隔符。在德国，人们用逗号写 0,5，所以在德国经常用分号用做列分隔符。如果你将一个 ROBO Pro CSV 文件输出到，比如 Microsoft Excel 有问题，可以试一下另外的分隔符。
- 在“列表数据类型(List Data type)”你可以选择发送或接收指令中携带的数据类型，可以是整数或浮点数。具体请参考 [13 章小数功能](#)。



- 在“**列表数据适用范围(List data life time)**”一栏，可以将列表变量设置为全局变量或者局部变量（见 [8.4.2 “局部变量”](#) 一节）。对于大列表（超过 100 个元素），建议设置为**全局**变量，因为全局变量比局部变量有更多的存储空间。

Cedutech®

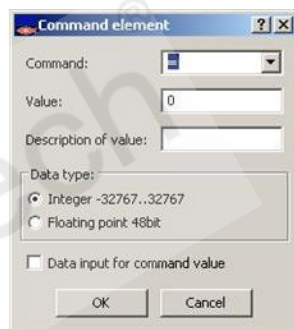
8.5 指令（级别 3）



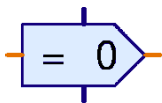
这一组的所有的程序模块都是指令模块。根据它们的应用，也可以是信息模块。指令模块执行的时候（控制流程进入模块的顶部的蓝色输入端），指令模块传递一条指令或者信息到连到它右部输出的模块。有各种各样的指令对所连接的模块有不同的作用，比如向左、向右、或者停止等。作为惯例，所连接的模块只能接受一些指令。每个程序模块接受的指令和那些指令的效果在各种程序模块旁边列出来了。大多数指令都附带着一个值。比如“向右”指令附带的，是指定的一个在 1 和 8 之间的速度。但另一方面，“停止”指令没有附加值。

指令模块的属性窗口

- 在“**指令(Command)**”一栏中，可以在可能的指令列表中选择想要的指令。
- 在“**值(Value)**”一栏中，可以输入随指令附带的数值。如果没有附带值，这一栏为空。
- 在“**值的描述(Description of value)**”一栏中，可以输入一个简短的表示文本（例如：X=或者 T=），显示在有附带值的指令模块中。这段文字应该说明包含了哪种类型的值。但是这一部分只是作为显示内容，没有其它的作用。
- 在“**数据类型(Data type)**”你可以选择发送或接收指令中携带的数据类型，可以是整数或浮点数。具体请参考 [13 章小数功能](#)。
- 在“**外部数据赋值(Data input for command value)**”一栏中，可以指定是否在指令模块的左边带一个橙色的用来附带数据的数据输入端。对于所有的指令模块，数值可以输入在指令模块内，也可以从指令模块左边的数据输入端读取。这样一来，比如对于一个电机，可以用一个带可调速的的伺服循环控制。



8.5.1 = （赋值）

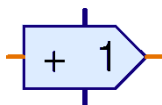


赋值指令“=”将一个数值分配给接收者。按照惯例，其经常用来将数值分配给变量，定时器变量，列表模块或者面板输出。

但是赋值指令“=”不仅可以由指令模块传递，也可由所有带数据输出的程序模块来传递。输出数据改变的时候，所有的模块传递“=”指令。例如，一个数字量输入模块，在输入端的传感器闭合的时候传递一个“=1”指令，而在传感器打开的时候传递一个“=0”指令。但是没有指令模块用来这么做。可以这么说，所有带数据输出的程序模块都有一个内置的赋值“=”指令。

所有的 ROBO Pro 程序模块数据输入端至少可以处理赋值“=”指令。这样，使得赋值“=”指令成为 ROBO Pro 中使用最频繁的指令。

8.5.2 + (加)

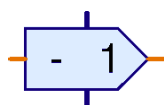


指令“+”可以传递到变量或者定时器变量来增加变量的值。指令“+”可以附带任何一个想要的值，并加到变量上。因为指令附带的值也可以为负，变量的值也可以用此指令来减少。见 [8.4.1“变量”](#) 一节和 [8.4.4“定时器变量”](#)

一节。

Cedutech®

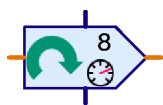
8.5.3 - (减)



指令“-”和上述的指令“+”比较相似。唯一的区别在于，指令所附带的值会从变量的值里面减去。

Cedutech®

8.5.4 向右

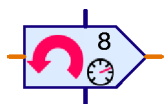


向右(Right)指令传递到一个电机输出模块来切换电机到顺时针方向。见 [8.7.4 “电机输出”](#) 一节。

速度值从 1 到 8。

Cedutech®

8.5.5 向左

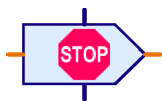


向左(Left)指令传递到一个电机输出模块来切换电机到逆时针方向。见 [8.7.4 “电机输出”](#) 一节。

速度值从 1 到 8。

Cedutech®

8.5.6 停止

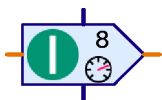


向右(**Right**)指令传递到一个电机输出模块来切换电机到顺时针方向。见 [8.7.4 “电机输出”](#) 一节。

没有值随“**Stop**”指令传递。

Cedutech®

8.5.7 开启

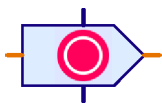


开启(On)指令传递到一个灯输出模块来将灯打开。见 [8.7.5 “灯输出”](#) 一节。“On”指令也可以传递到电机输出模块，相当于“**向右**”指令。然而，对于电机输出用“**向右**”指令更好一些，因为可以直接辨识电机旋转的方向。

其值为亮度或强度，从 1 到 8。

Cedutech®

8.5.8 关闭

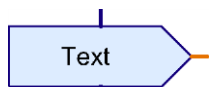


关闭(Off)指令传递到一个灯输出模块来将灯关闭。见“灯输出”一节。“**Off**”指令也可以传递到电机输出模块，相当于“**停止**”指令。

没有值随“**off**”指令传递。

Cedutech®

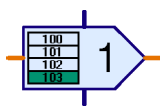
8.5.9 文本



文本(Text)指令是一条特殊的指令，由于它不是传递一条带数值的指令，而是你选择的一个文本，到所连接的模块。然而，只有一个程序模块可以处理“文本”指令，它是面板中的文本显示指令。可以在 [9.1.2 “文本显示”](#) 一节找到相关信息。

Cedutech®

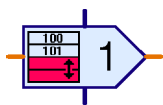
8.5.10 添加数值



添加数值(Append value)指令是针对列表的一条特殊的指令。见 [8.4.5 “列表”](#) 一节。这条指令附带着一个数值，用来添加到列表的末尾。如果序列已经满了，则会忽略这条指令。

Cedutech®

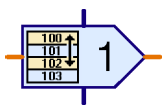
8.5.11 删除数值



删除数值>Delete values)指令是针对列表的一条特殊的指令。见 [8.4.5 “列表”](#) 一节。用这条指令，可以删除序列末尾的任何数值。想要的号码作为数值随指令附带，如果这个值大于列表中元素的数量，序列中所有的数会被删除。为了完全删除一个序列，可以传递一个带最大值“32767”的“Delete”指令。

Cedutech®

8.5.12 交换数值



交换数值(Exchange values)指令是针对列表的一条特殊的指令。见 [8.4.5 “列表”](#) 一节。用这条指令，列表中所有的元素都可以和第一个元素交换。要和第一个元素交换的元素号作为一个数值随指令附带。**注意：**序列

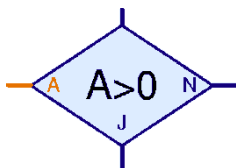
第一个元素的编号为 0。如果指令附带的值不是一个有效的序号，列表模块会忽略此指令。

8.6 比较，等待 ... (级别 3)

这一组的程序模块都是用作程序控制的分支或者延迟程序的持续运行。

Cedutech®

8.6.1 判断 (带数据输入)



这个程序判断模块的左边有一个橙色的数据输入端“A”。通过这个端口，可以读入一个经常来自输入模块（见 [8.7 章节“控制板输入/输出”](#)）的数值。数据输入端“A”可以和变量、定时器变量或运算符的输出相关联（见 [8.8 章节“运算符”](#)）。

模块将来自数据输入端“A”的数值和一个固定但可自由定义的值比较。根据比较是否保持，决定模块的分支以“Y”或者“N”为出口。

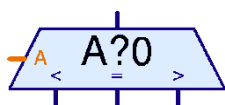
判断模块的属性窗口

- 在“条件(Condition)”这一栏，在右边的区域，可以输入用来和输入值 A 作比较的数值。通常的比较运算符对于这一比较是有效的。
- 在“数据类型(Data type)”你可以选择发送或接收指令中携带的数据类型，可以是整数或浮点数。具体请参考 [13 章小数功能](#)。
- 在“交换 Y/N 分支位置(Swap Y/N branches)”一栏，可以交换分支中“Y”和“N”出口的位置。一般情况下，“Y”出口在下部，“N”出口在右边。但有时候将“Y”出口放到右边会更实用，按一下“Swap 1/0 branches”，点击 OK 关闭对话框时，这两个出口就互换了。

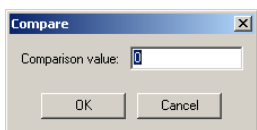


最常用的比较是 $A > 0$ 。意味着如果在数据输入端 A 出现的数值大于 0，那么控制分支就以 Y 为出口。例如，仅传递 1 或者 0 值的数字量输入，就可以用这个方法评估。但是，定时器变量和其它很多的数值也同样可以用比较式 $A > 0$ 来评估。

8.6.2 与固定值作比较



用程序模块“与固定值作比较(Comparison with fixed value)”，数据输入端 **A** 的数值可以和一个固定的，但可以自由定义的数值作比较。根据在数据输入端 **A** 出现的数值大于、小于或者等于固定值，那么控制分支就以比较模块的右边，左边或者中间为出口。作为一个惯例，经常将变量和列表的输出连接到数据输入端 **A**。比较模块可以用两个判断模块来代替。然而，在很多情况下，如果只需一个模块的话可理解性会大得多。

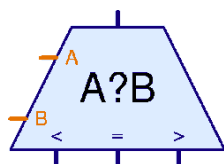


注意：这个模块不适用于浮点数，因为浮点数由于在取整时会产生比较错误，所以是否完全相同就不得而知。但是可以使用判断模块中区别大小，见 [8.6.1 “判断（带数据输入）”](#)。

比较模块属性窗口

在“比较值(Comparison value)”一栏，可以输入一个和输入端 **A** 的值进行比较的的常数。

8.6.3 比较

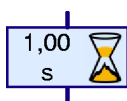


使用“**比较(Compare)**”程序模块，数据输入端 **A** 和 **B** 的数值可以相互比较。根据 **A** 小于 **B**，**A** 大于 **B**，或者 **A** 等于 **B**，模块分支以左边，右边或者中间作为出口。这个模块最常用于将一个目标值和一个实际的值作比较。根据目标值值和实际值的相对关系，比如电机可以左转、右转或者停止。

“**Compare**”程序模块没有选项可以设置，所以也没有属性窗口。

注意：这个模块不适用于浮点数，因为浮点数由于在取整时会产生比较错误，所以是否完全相同就不得而知。但是可以使用判断模块中区别大小，见 [8.6.1 “判断（带数据输入）”](#)。

8.6.4 延时

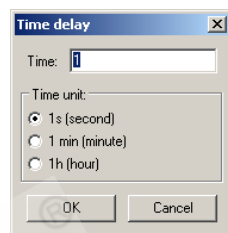


用这个模块，一个“**延时(Time delay)**”可以编进程序流程中。当轮到其执行时，时间延时就开始了。当延时时间一到，程序就继续执行。也可参见 [3.6.1 “延时”](#) 一节。

延时模块的属性窗口：

在“**时间(Time)**”一栏，可以输入所延迟的时间。你甚至可以使用小数，例如 1.23。

在“**时间单位(Time unit)**”一栏，可以选择秒，分钟或者小时作为时间单位。这里的时间单位，并不像定时器变量中，它对延时的精度并没有影响。一个 60 秒的时间延迟和一分钟的时间延迟动作是完全一样的。



8.6.5 等待...

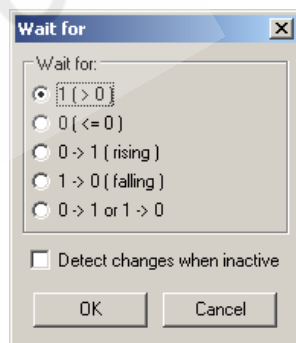


“等待(Wait for...)”程序模块可以阻止程序的执行，直到发生一个变化或者在模块的数据输入端达到一个特定的状态。模块有五种变化状态：模块左端等待直到输入端的值增加。对于此用途，并不光是从 0 到 1 的变化，而且是任何一种上升，比如从 2 到 3，也计在内。第二种模块等待直到输入端的数值减少。在中间的模块则等待变化，不管是任何方向。第三种模块经常用在脉冲齿轮上。第四种和第五种模块等待的不是变化，而是输入端的状态的是(>0)或者非(<=0)。如果已经达到了相应的状态，则模块不再等待。另一方面，前三种模块总是等待直到检测到输入端的变化。

等待模块的属性窗口

在“等待(Type of change)”一栏，可以在上述五种功能中作选择。

如果选择了“触发变化检测(Detect changes when not active)”，模块在其不应当执行的时候，也会检测信号变化的发生。在此情况下，模块保存了最近的值。当模块再次执行的时候，如果值在间歇期已经按正确的方式变化了，程序就可以立即继续执行。在这种方式下，丢失信号变化的可能性就较小了，因为程序去完成其它任务了。



8.6.6 脉冲计数器

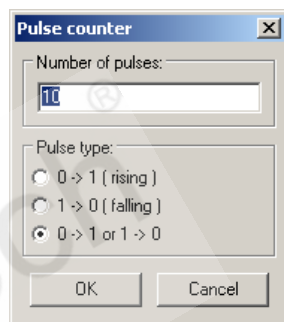


这个程序模块在继续执行程序之前，要等待一个可定义的，来自左边数据输入端的脉冲输入数。这对于用脉冲齿轮的简单的定位任务是非常实用的。要更精确的定位，比如用变量值，必须用带变量的子程序。

脉冲计数器的属性窗口

- 在“脉冲数量(Number of pulses)”一栏，可以输入在程序继续执行前，要等待的脉冲数。
- 在“脉冲类型(Pulse type)”一栏，可以选择三种形式的脉冲中的任一种：**0-1**，**1-0**或者两者皆是。

在模块未执行时识别变化的可能性，如“Wait for ...”模块所能实现的那样，在这个模块中无效。



8.7 接口板输入/输出

这一组的程序模块包含所有的输入和输出模块。如何使用这些模块可以详见 [4.4 “级别 3：变量，面板和指令”](#) 一节。

Cedutech®

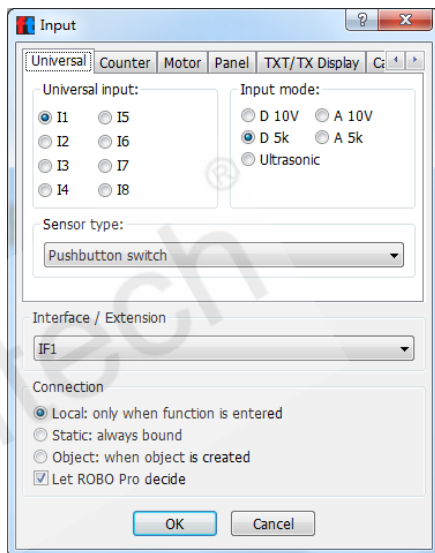
8.7.1 通用输入



TX 控制板的 8 路通用输入 **I1** 到 **I8** 可以作为数字量或模拟量输入。你可以将软件中的按钮(buttons)或是任意慧鱼传感器与其相连。

通用输入模块的属性窗口：

- 在“通用输入(Universal input)”一栏，可以选择所用的接口板输入端号。扩展接口板上的输入端可以在“接口板 / 扩展板 (Interface / Extension)”中选择。
- 在“传感器类型(Sensor type)”一栏，可以选择连接到输入端的传感器。
- 在“输入模式(Input mode)”一栏，你可以选择输入量是数字量或模拟量，具体是对于电压，电阻或是超声波距离传感器信号的。详情见 12.5 “通用输入，长安其类型和输入模式”章节。ROBO Pro 软件可以自动根据选择的传感器类型决定输入模式。例如，对于光敏晶体管，软件自动将输入模式调整为数字量 5kOhm(D 5k)。这样，你可以将光敏晶体管和透镜灯配合，作为光幕(light barrier)使用，当有遮挡时信号为 0，正常信号为 1。然而对于光敏电阻，就需要使用模拟量 5kOhm(A 5k)的输入模式，这样可以区分出光线的强弱。
- 在“控制板/扩展板(Interface/extension)”一栏，可以选择使用主控制板或扩展板的输入端口。详细信息可以参见[第 7 章“控制多个控制板”](#)。
- 在“连接(Connection)”一栏，你可以选择该模块是否总是与其他模块接通，或是在包含该模块的子程序被调用时才接通。这或与子程序输入端连接的全局变量不同。



经更仔细的查看，对于所有的输入类型都只有一种类型的程序模块。你随时可以通过属性窗口顶部的标签来多次切换输入端。这种功能在开关输入和面板输入间切换时特别有用。

Cedutech®

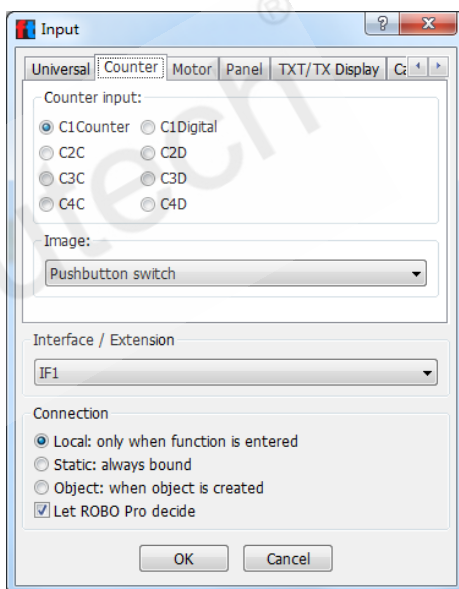
8.7.2 计数输入



除了 8 路通用输入端，TXT 和 TX 控制板还提供 4 路计数输入端 C1-C4，你可以将数字量传感器或是带编码器电机中的编码器与之相连。对于每一路计数输入，例如 C1，在 ROBO Pro 中有两种计数输入模式 C1C 和 C1D。C1D 模式与普通的数字量输入相似。C1C 模式用于计数输入，记录 C1 端口的脉冲数，可以通过复位(reset)命令复位。除非对应电机接口上没有连接带编码器电机（例如 M1 端口对应 C1 端口），否则这些端口只能为带编码器电机服务。

模拟量输入模块的属性窗口：

- 在“计数输入 (Counter input)”一栏，可以选择作为计数端或数字量输入端使用。
- 在“类型(Image)”一栏，可以选择连接到输入端的传感器的图示。
- 在“控制板 / 扩展板 (Interface/extension)”一栏，可以选择使用主控制板或扩展板的输入端口。详细信息可以参见[第 7 章“控制多个控制板”](#)。
- 在“连接(Connection)”一栏，你可以选择该模块是否总是与其他模块接通，或是在包含该模块的子程序被调用时才接通。这或与子程序输入端连接的全局变量不同。



对所有的输入，ROBO Pro 都用一个单一的模块，通过顶部的标签可以在各输入类型间切换。然而，为了简化起见，在模块窗口中各种输入模块都是分开选择的。

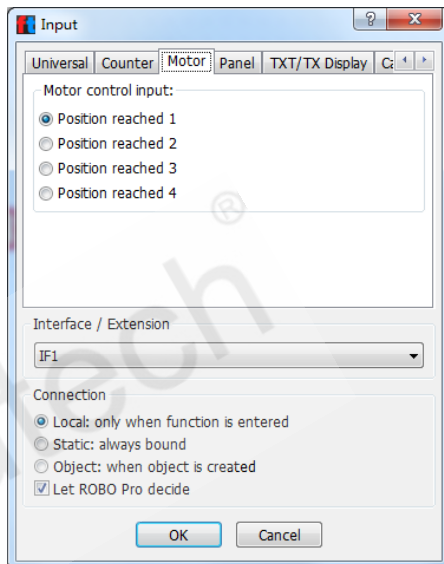
8.7.3 电机到位输入

M1E
IF1

这些输入不是实际输入，而是内部变量输入，用于控制带编码器的电机。你可以在 [12.6.2 “级别 3 中的扩展电机控制”](#) 了解到更多。

电机到位输入模块的属性窗口：

- 在“电机到位输入 (Motor control input)”一栏，可以选择查询哪一个端口的电机到位信号。
- 在“控制板 / 扩展板 (Interface/extension)”一栏，可以选择使用主控制板或扩展板的输入端口。详细信息可以参见 [第 7 章“控制多个控制板”](#)。
- 在“连接(Connection)”一栏，你可以选择该模块是否总是与其他模块接通，或是在包含该模块的子程序被调用时才接通。这和其他子程序输入端连接的全局变量不同。



8.7.4 电机输出



“电机输出(Motor output)”模块可以控制控制板的四路双向电机输出端。一个电机输出通常使用两个控制板接口，而灯输出只用一个接口。在 [8.1.6 “电机输出”](#) 和 [8.1.8 “灯输出”](#) 章节，你可以找到电机和灯输出之间区别的更多内容。

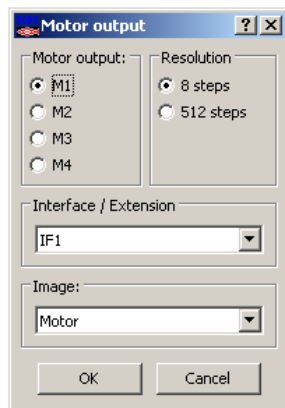
必须用一个指令模块传递一个指令到电机输出来控制输出。电机输出模块可以用下列指令来处理：

指令	值	动作
向右	1 to 8	电机以速度 1 到 8 顺时针旋转
向左	1 to 8	电机以速度 1 到 8 逆时针旋转
停止	无	电机停止
打开	1 to 8	和向右一样
关闭	无	和停止一样
=	-8 to 8	值 -1 to -8: 电机顺时针旋转 值 1 to 8: 电机逆时针旋转 值 0: 电机停止

另外，电机输出模块可以接收扩展电机控制（同步，距离和复位），详见 [12.6.2 “级别 3 中的扩展电机控制”](#) 章节。

电机输出模块的属性窗口

- 在“电机输出(Motor output)”一栏，可以选择使用的接口板输出端口。也可以在 **Interface / Extension** 一栏中选择扩展接口板的输出端口。
- 在“精度(Resolution)”一栏，可以选择控制的精度，可以是 1-8 的 8 级调速，或是 1-512 的 512 级调速。
- 在“控制板/扩展板(Interface/extension)”一栏，可以选择使用主控制板或扩展板的输入端口。详细信息可以参见[第 7 章 “控制多个控制板”](#)。



- 在“**类型(Image)**”一栏，可以选择接到输出接口的负载的图示。大多数情况下是一个电机，但也可以接一个电磁铁，电磁阀或者灯到电机输出。

Cedutech®

8.7.5 灯输出



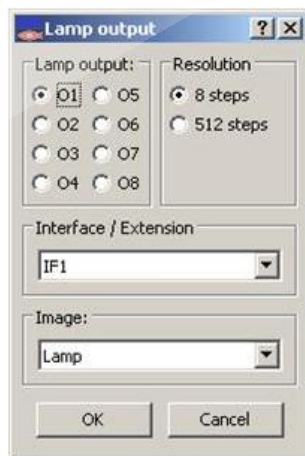
“灯输出(Lamp output)”模块可以控制控制板的 8 路单极灯输出 O1-O8 之一。灯输出只用了控制板的一个输出端口，灯的另一根线接到了控制板的接地端。在这种接线方式下，负载灯只能打开或者关闭，你无法改变它的极性。在 [8.1.6 “电机输出”](#) 和 [8.1.8 “灯输出”](#) 章节，你可以找到电机和灯输出之间区别的更多内容。

必须用一个指令模块传递一个指令到灯输出来控制输出。灯输出模块可以用下列指令来处理：

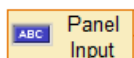
指令	值	动作
打开	1 到 8	灯打开到 1 到 8 之间的一种亮度
关闭	无	灯被关闭
=	0 to 8	值 1 to 8: 灯打开 值 0: 灯关闭

灯输出模块的属性窗口：

- 在“灯输出(Lamp output)”一栏，可以选择使用的接口板输出端口。也可以在 **Interface / Extension** 一栏中选择扩展接口板的输出端口。
- 在“精度(Resolution)”一栏，可以选择控制的精度，可以是 1-8 的 8 级调速，或是 1-512 的 512 级调速。
- 在“控制板/扩展板(Interface/extension)”一栏，可以选择使用主控制板或扩展板的输入端口。详细信息可以参见[第 7 章“控制多个控制板”](#)。
- 在“类型(Image)”一栏，可以选择接到输出接口的负载的图示。大多数情况下是一个灯，但也可以接一个电磁铁，电磁阀或者电机到灯输出。但是接到灯输出的电机只能单向旋转。



8.7.6 面板输入

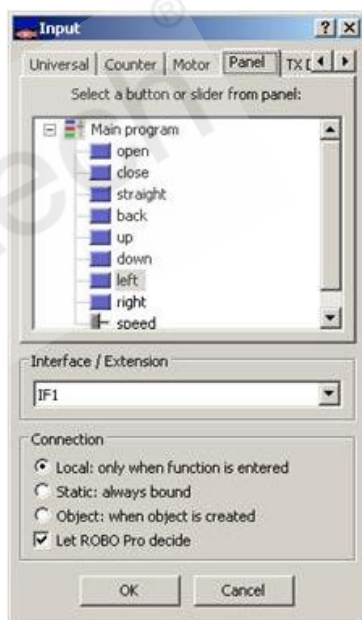


ROBO Pro 提供了为你的模型设计自己的面板的可能。你可以在[第 9 章“面板模块和面板：概览”](#)学到更多相关的内容。这样可以使你更方便地在电脑上控制你的模型。按钮、滑动条和数据输入模块可以在面板中使用。这些模块的状态可以通过“**面板输入(Panel input)**”模块来查询。按钮返回一个“0”或“1”值，滑动条返回一个用户可定义的范围中的值(默认为 0 到 100)。

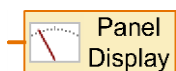
面板只能在线模式下使用。更多信息可以参考[3.7“在线和下载模式的差别”](#)一节。

面板输入模块的属性窗口：

- 一块面板和主程序或者子程序相关联。面板模块在相关的程序名字下列出。如果你还没有定义任何的面板模块，那么列表中没有任何模块。因此，必须在将一个面板输入和一个面板模块关联之前，先设计面板。
- 在面板输入的情况下，在“**控制板/扩展板(Interface / Extension)**”一栏的选项被忽略，因为这里我们不处理控制板上的实际输入。



8.7.7 面板输出

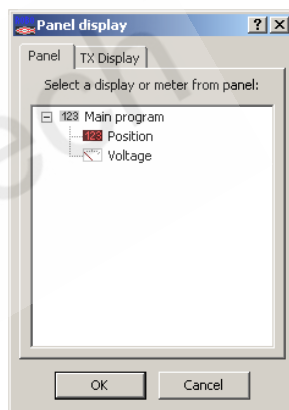


ROBO Pro 提供了为你的模型设计自己的面板的可能。你可以在[第 9 章“面板模块和面板：概览”](#)学到更多相关的内容。除了用按钮和其它输入模块来控制你的模型，也可以在你的面板中插入显示模块。在这些显示模块中，可以显示，例如机器人的轴的坐标或者极限开关的位置。你可以通过在你的程序中插入一个“**面板输出(Panel output)**”模块来显示值，连接一个变量、一个模拟量输入或者一个指令模块。

面板只能在在线模式下使用。更多信息可以参考[3.7“在线和下载模式的差别”](#)一节。

面板输出模块的属性窗口：

- 面板属于每一个主程序或者子程序。面板模块在相关的程序名字下列出。如果你还没有定义任何的面板模块，那么列表中没有任何模块。因此，必须将在一个面板输入和一个面板模块关联之前，先设计面板。



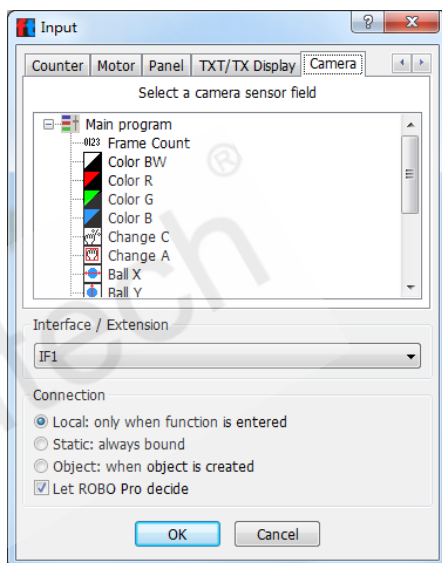
8.7.8 摄像头输入

4.x 版本或更高版本的 ROBO Pro 软件支持慧鱼 USB 摄像头。在摄像头窗口（详见 [11 章](#)），你可以添加摄像头识别区域，用以识别颜色，运动物体，线条，或是小球。你可以通过“**摄像头输入(Camera Input)**”查询识别区域的返回值。

摄像头输入模块的属相窗口：

- 摄像头输入属性窗口中包含一系列输入选项，根据选择用于识别区域的模块不同，对应的选项会呈现在属性窗口中。如果没有添加任何识别区域，这个列表就是空的。

你可以在 [11.3](#) 章节中了解到更多关于窗口中输入类型的信息。



8.8 运算符

这一组的所有程序模块称之为运算符。运算符有一个或多个橙色数据输入端，从数据输入端来的数值由运算符组合得出一个新值，此新值由运算符的输出端用一个“=”指令传递。

运算符的属性窗口：

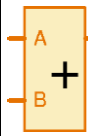
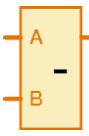
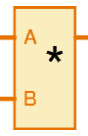
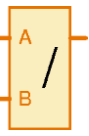
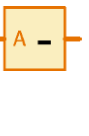
所有的运算符使用相同的属性窗口。通过属性窗口，你甚至可以将一个运算符转换为另一个运算符。

- 在“**运算(Operation)**”一栏，可以设置运算符如何来组合它的输入。各个单个的功能将在以下两节来介绍。
- 在“**输入端数量(Number of inputs)**”一栏，可以设置运算符中参与运算的数的个数。
- 在“**数据类型(Data type)**”你可以选择发送或接收指令中携带的数据类型，可以是整数或浮点数。具体请参考 [13章小数功能](#)。



8.8.1 算数运算符

ROBO Pro 有四种基本的算数运算符。带有两个输入端，符号如下：

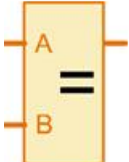
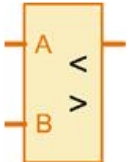
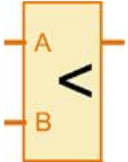
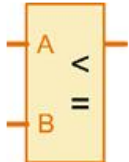
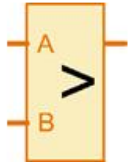
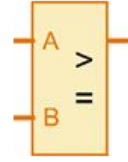
				
加	减	乘	除	取负
$A + B$	$A - B$	$A * B$	A / B	$- A$

如果“**减(Minus)**”运算符有超过两个输入端，所有后来的输入值从 A 输入端的数值中减去。如果“**Minus**”运算符只有一个输入端，则运算符改变输入值的符号。

如果“**除(Divided by)**”有两个以上的输入端，则输入端 A 的数值被所有其它值相除。

8.8.2 比较运算符（关系运算符）

ROBO Pro 中共有 6 中关系运算符：

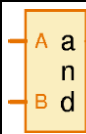
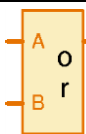
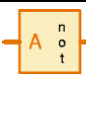
					
相等	不等	小于	小于等于	大于	大于等于
$A=B$	$A \neq B$	$A < B$	$A \leq B$	$A > B$	$A \geq B$

如果比较结果为真，输出端输出 1，否则输出 0。输出值为整数，即使参与比较的数位浮点数。

除去不等于运算符只能有两个数参与比较，其他运算符都可以有多个数参与比较。例如，只使用一个运算符，就可以决定一个数是否在给定区间内，若 $A=0$ ， $B=1$ ， $C=2$ ，则 A,B,C 顺序参与小于运算符，则输出值为 1；若 $B=9$ ，其余不变，则输出值为 0。

8.8.3 逻辑运算符

ROBO Pro 有三种逻辑运算符。

		
与	或	非
$A > 0$ 与 $B > 0$	$A > 0$ 或 $B > 0$	$A \leq 0$

逻辑运算符将一个大于零的值看作 **yes (是)** 或者 **true (真)**， 并把一个小于等于零的数看作 **no (非)** 或者 **false (假)**。数字量输入返回一个值“0”或者“1”，这样“0”被看作 **false (假)**，而“1”看作 **true (真)**。

如果所有的输入值都为真(>0)， “**与(And)**” 运算符传递一个附带值为 1 的赋值指令，到连结在其输出的模块。否则模块将传递一个附带值为 0 的赋值指令。

如果至少一个输入值为真 (>0)， “**或(Or)**” 运算符将传递一个附带值为 1 的赋值指令，到连结在其输出的模块。否则模块将传递一个附带值为 0 的赋值指令。

如果其输入值为假 (≤ 0)， “**非(Not)**” 运算符将传递一个附带值为 1 的赋值指令，到连结在其输出的模块。否则模块将传递一个附带值为 0 的赋值指令。

逻辑运算符的功能也可以用几个判断模块来仿效，但是用运算符来组合几个输入的可读性强的多。

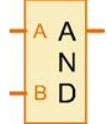
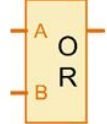

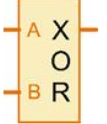
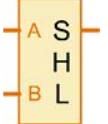

8.8.4 位运算符

ROBO Pro 软件中每个变量都由 16 位二进制代码组成，每一位可以是 0 或 1，具体对应关系如下：

位	二进制数	对应关系	十进制数
0	0000000000000001	2^0	1
1	0000000000000010	2^1	2
2	0000000000000100	2^2	4
3	0000000000001000	2^3	8
4	0000000000010000	2^4	16
5	0000000000100000	2^5	32
6	0000000001000000	2^6	64
7	0000000010000000	2^7	128
8	0000000100000000	2^8	256
9	0000001000000000	2^9	512
10	0000010000000000	2^{10}	1024
11	0000100000000000	2^{11}	2048
12	0001000000000000	2^{12}	4096
13	0010000000000000	2^{13}	8192
14	0100000000000000	2^{14}	16384
15	1000000000000000	2^{15}	-32768

例如，十进制数 3 的 0 位和 1 位均为 1，则 $2^0+2^1=3$ ，位运算符与逻辑运算符类似，只不过是每一位都进行逻辑运算。例如 3 位与 (AND) 6 等于 2，其中 $3=2^0+2^1$ ， $6=2^1+2^2$ ，每一位分别进行逻辑与运算后为 $2^1=2$ ，所以结果为 2。请注意十进制数 32768，最高位为 1，在 ROBO Pro 中代表错误或空值，要产生这个值，在输入时不写入任何数字即可。

以下为各个位运算符：

					
位与	位或	位非	异或	左移	右移
结果为 A 与 B 按位与运算得到的数	结果为 A 与 B 按位或运算得到的数	结果为 A 按位非运算得到的数	A 与 B 每一位不相同取 1，相同取 0，得到的数	A 按位向左移 B 个数位得到的数	A 按位向右移 B 个数位得到的数

8.8.5 函数

函数模块与运算符类似，但是只有一个输出，函数模块包括三角函数，开平方，指数，对数。

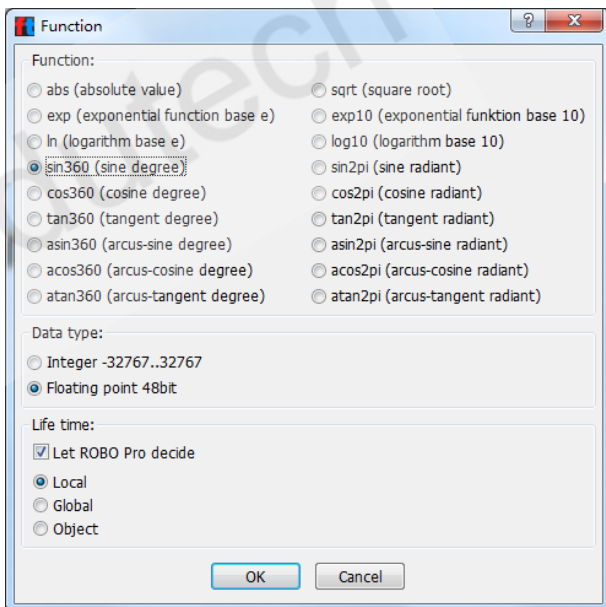
注意 1: 许多情况下，运算函数非常困难。由于 TX 控制器需要确保执行指令时，每秒要进行 1000 次扫描，而函数运算则被限制在一个指令内。橙色数据线总是在处理数据，并没有断开。因此，在同一条橙色数据线上，不能连接太多函数运算。

注意 2: ROBO Pro 并不用扩展精度的运算去计算函数。所以，结果的精度通常比标准 48 位浮点数格式少 2 位，结果的精度由 ROBO Pro 预估，并保存于结果中。

函数模块的属性窗口：

所有函数均使用相同的属性窗口。

- 在“**函数(Function)**”一栏，你可以选择使用的数学函数，单独的函数会在下面解释。
- 在“**数据类型(Data type)**”你可以选择发送或接收指令中携带的数据类型，可以是整数或浮点数。具体请参考 [13 章小数功能](#)。



基础函数

Abs	绝对值：对输入值取正，例如输入-3.2，得到 3.2
Sqrt	开平方根：对输入值开平方，例如输入 2.0，得到 1.4142...

指数与对数函数

Exp	以 e 为底的指数，输入 x，得到 e^x
Exp10	以 10 为底的指数，输入 x，得到 10^x ，例如输入 2.0，得到 100.0
Log	以 e 为底的对数，通常写作 \ln
Log10	以 10 为底的对数，例如输入 1000，得到 3.0

三角函数和反三角函数

所有三角函数和反三角函数都有两种角度单位，度（一周 360 度）和弧度（一周 2π 弧度）。

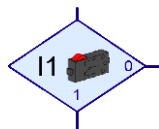
Sin360/sin2pi	求正弦
Cos360/cos2pi	求余弦
Tan360/tan2pi	求正切
Asin360/asin2pi	求反正弦
Acos360/acos2pi	求反余弦
Atan360/atan2pi	求反正切

8.9 ROBO 接口板

Cedutech®

8.9.1 数字量分支

根据某一个数字量输入 **I1-I8** 的状态，在一个或者两个方向上你可以直接用此分支来编程控制。比如，数字量输入的某个传感器闭合 (=1)，则程序分支走“**1**”出口。反之，如果输入断开，则程序分支走“**0**”出口。



如果鼠标右键点击模块，会有属性窗口打开：

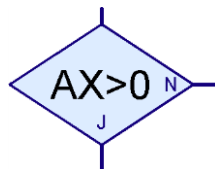
- 按钮“**I1**”到“**I8**”允许选择某一个要查询的接口板输入。
- 在“**接口板/扩展板(Interface/extension)**”一栏，可以选择使用主控制板或扩展板的输入端口。详细信息可以参见[第 7 章“控制多个控制板”](#)。
- 在“**类型(Image)**”一栏,可以选择连接到输入端的传感器图示。按钮传感器是最常用的数字量输入形式，但也经常用光电传感器和干簧管。
- 在“**交换 1/0 分支位置(Swap Y/N branches)**”一栏，可以交换分支中“**1**”和“**0**”出口的位置。



一般情况下，“**1**”出口在下部，“**0**”出口在右边。但有时候将“**1**”出口放到右边会更实用，按一下“**Swap Y/N branches**”，点击 **OK** 关闭对话框后，这两个出口就互换了。

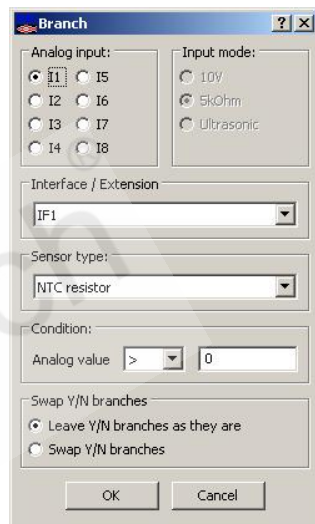
8.9.2 模拟分支

除了数字量输入, ROBO 接口板有 6 路模拟量输入: 2 路阻抗输入 AX 和 AY, 2 路电压输入 A1 和 A2, 以及 2 路距离传感器输入 D1 和 D2。用此模块可以将模拟量输入值和固定值进行比较, 根据比较的结果, 来确分支的“**Yes(Y)**”或者“**No(N)**”出口。



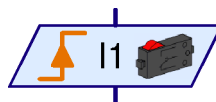
如果鼠标右键点击模块, 会有属性窗口打开:

- 在“**模拟量输入(Analog input)**”一栏, 可以选择某一个要查询的接口板输入。所有的模拟量输入都返回一个 0—1023 的值。详细信息可以参见第 7.6.2 节各种模拟量输入方面的信息。
- 在“**接口板/扩展板(Interface/extension)**”一栏, 可以选择使用主控制板或扩展板的输入端口。详细信息可以参见[第 7 章“控制多个控制板”](#)。
- 在“**条件(condition)**”一栏, 可以选择一个比较算式, 比如小于 (<) 或者大于 (>), 并输入比较值。比较值应该在 0 和 1023 之间。当在线模式下启动一个含模拟分支的程序时, 当前的模拟值会显示出来。
- 在“**交换 1/0 分支位置(Swap Y/N branches)**”一栏, 可以交换分支中“1”和“0”出口的位置。一般情况下, “1”出口在下部, “0”出口在右边。但有时候将“1”出口放到右边会更实用, 按一下“**Swap Y/N branches**”, 点击 **OK** 关闭对话框后, 这两个出口就互换了。



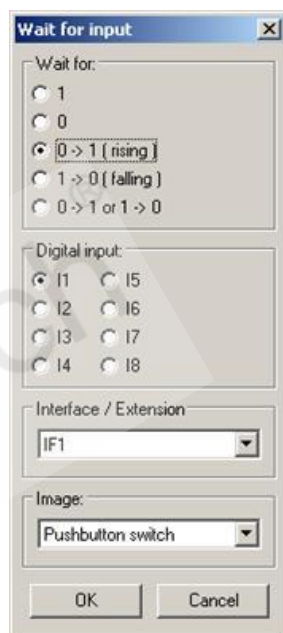
8.9.3 输入等待

“输入等待”模块的功能是，等待直到接口板的某个输入变为特定状态或者其状态由某一特定方式改变。

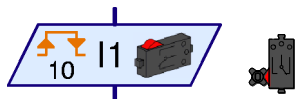


如果鼠标右键点击模块，会有属性窗口打开：

- 在“等待(Wait for)”一栏，可以选择信号变化的类型或者所等待的信号状态。如果你选择 1 或者 0，模块一直等待，直到输入信号闭合(1)或者打开(0)。如果你选择 0 -> 1 或者 1 -> 0，模块一直等待，输入信号状态从打开到闭合变化(0->1) 或者从闭合到打开 (1->0)。最后一种情况是，模块一直等待，直到输入信号状态变化。而不管是从打开到闭合，反之亦然。为帮助你更好地理解，请见 [3.6 “其他程序模块”](#) 章节，如何用判断模块来模拟这个模块。
- 在“模拟量输入(Digital input)”一栏，可以确定读取接口板中 I1 - I8 的任一输入信号。
- 在“接口板/扩展板(Interface/extension)”一栏，可以选择使用主控制板或扩展板的输入端口。详细信息可以参见 [第 7 章“控制多个控制板”](#)。
- 在“类型(Image)”一栏，可以选择连接到输入端的传感器图示。微动传感器是最常用的数字量输入形式，但也经常用光电传感器和干簧管。



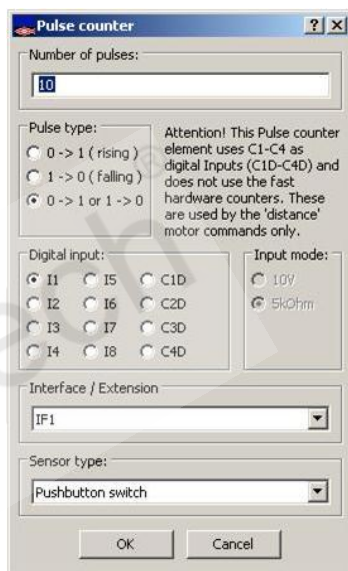
8.9.4 脉冲计数器



许多的慧鱼模型都用到了脉冲齿轮。这些齿轮每一圈碰触传感器四次。使用这些脉冲齿轮，可以使电机运行精确定义的圈数，而不是一段给定的时间。为了实现此目的，必须对接口板输入端的脉冲进行计数。这样，由“**脉冲计数器**”模块来等待一个用户自定义的脉冲数。

如果鼠标右键点击模块，会有属性窗口打开：

- 在“**脉冲类型(Pulse type)**”一栏，可以选择所要计数的脉冲类型。如果选择 **0 -> 1**（上升沿），模块一直等待输入的状态从打开变为闭合(0->1)，跳变的次数可以“**脉冲数量(Number of pulses)**”中定义。如果选择 **1 -> 0**（下降沿），模块一直等待输入的状态从打开变为闭合(1->0)，跳变的次数同样在“**脉冲数量(Number of pulses)**”中定义。然而，对于脉冲齿轮，第三种可能性更常用：模块对 **0 -> 1** 和 **1 -> 0** 的变化都进行计数，这样，脉冲齿轮每转一圈可计得 8 个脉冲。
- 在“**数字量输入(Digital input)**”一栏，可以选择读取接口板中 **I1 - I8** 的任一输入信号。
- 在“**接口板/扩展板(Interface/extension)**”一栏，可以选择使用主控制板或扩展板的输入端口。详细信息可以参见[第 7 章“控制多个控制板”](#)。
- 在“**类型(Image)**”一栏，可以选择连接到输入端的传感器图示。“微动开关”传感器是最常用的数字量输入形式，但也经常用光电传感器和干簧管。



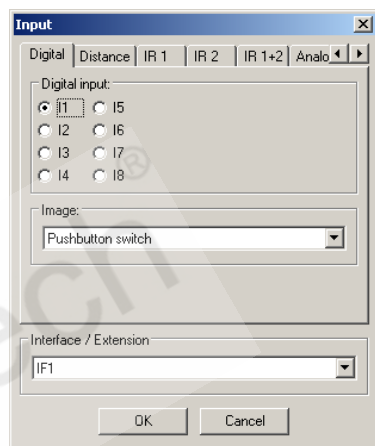
8.9.5 数字量输入



数字量输入 **I1** 到 **I8** 的值可以用“Digital input”模块来查询。如果接口板上输入端口电气上是闭合的，则数字量输入模块的橙色连接上会返回一个数值“1”，否则会返回一个数值“0”。

数字量输入的属性窗口：

- 在“数字量输入(Digital input)”一栏，可以选择所用的接口板输入端号。扩展接口板上的输入端可以在“接口板/扩展板(Interface / Extension)”中选择。
- 在“类型(Image)”一栏，可以选择连接到输入端的传感器图示。大多数情况下是一个“微动开关(mini-push-button)”。“干簧管(reed contact)”是一种可以检测磁场的传感器。虽然光电开关事实上是一种模拟的传感器，但也可以接到数字接口。你可以将一个带透镜的灯作为光束，和连接到输入端的光电开关一起使用。其在光束中断时为(=0) 或者未中断(=1)。另一方面，如果将光电开关接到一个模拟量输入端，可以用来区分在明亮和黑暗之间的许多阶段。
- 在“接口板/扩展板(Interface/extension)”一栏，可以选择使用主控制板或扩展板的输入端口。详细信息可以参见[第 7 章“控制多个控制板”](#)。



经更仔细的查看，对于所有的输入类型都只有一种类型的程序模块。你随时可以通过属性窗口顶部的标签来多次切换输入端。这种功能在开关输入、IR 输入和面板输入间切换时特别有用。

8.9.6 模拟量输入



模拟量输入端的值可以通过“**模拟量输入(Analog input)**”模块来进行查询。和数字量输入返回“0”和“1”值不同，模拟量输入可以分辨连续的输入。所有的模拟量输入返回一个“0 到 1023”之间的值。然而，ROBO 接口板有各种不同的模拟量输入接口来测量不同的物理量。具体来说，模拟量输入可以适应用来测量电阻，测量电压和测量距离的不同的传感器。

输入端	输入类型	测量范围
A1, A2	电压输入	0-10,23V
AX, AY	电阻输入	0-5,5kΩ
D1, D2	距离传感器输入	ca. 0-50cm
AV	电源电压	0-10V

通常的慧鱼传感器中，温度传感器和光电传感器将被测量（温度和光强度）转换成一个电阻值。所以必须将这些传感器接到 **AX** 和 **AY** 输入端。电压输入端 **A1** 和 **A2** 设计用来连接所有产生 0 到 10V 电压的传感器。

在 ROBO 接口板上没有针对 **AV** 输入的端口。它通常和接口板的电源电压相关联。这样一来，比如可以监控电池的电压，在电池耗尽之前可以将模型撤离。

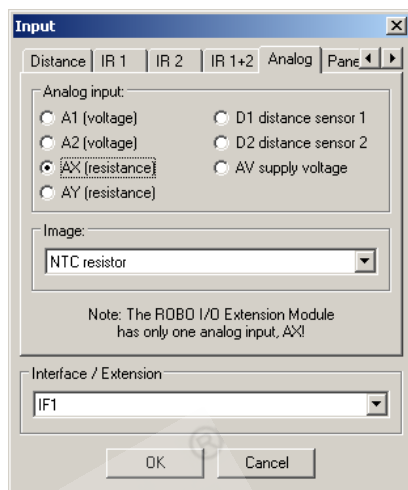
距离传感器输入端 **D1** 和 **D2** 可以接到慧鱼特殊传感器，可以测量（比如到障碍物）的距离。

智能接口板只有两个模拟量输入端 **EX** 和 **EY**。这些相当于 ROBO 接口板的 **AX** 和 **AY**。其它模拟量输入端无法在智能接口板上用。

模拟量输入属性窗口：

- 在“**模拟输入(Analog input)**”一栏，可以选择想要用的模拟量输入端。
- 在“**类型(Image)**”一栏，可以选择连接到输入端的传感器的图示。
- 在“**接口板/扩展板(Interface/extension)**”一栏，可以选择使用主控制板或扩展板的输入端口。详细信息可以参见[第 7 章“控制多个控制板”](#)。

对所有的输入，ROBO Pro 都用一个单一的模块，通过顶部的标签可以在各输入类型间切换。这一点在模拟量输入的属性窗口中可以看得更清楚了。然而，为了简化起见，在模块窗口中各种输入模块都是分开选择的。



8.9.7 红外线输入



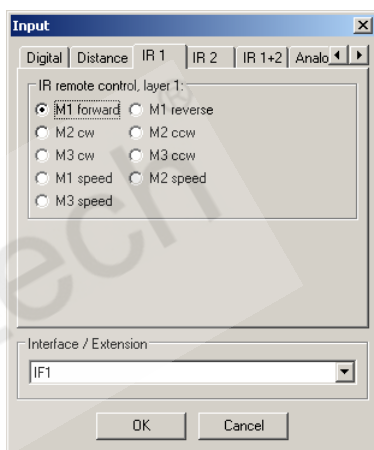
ROBO 接口板有一个内置的红外线接收器，可以用慧鱼**远红外控制组**(30344)中的手持式发射器来控制。

手持式发射器非常有用，不仅可以遥控，还可以做为模型控制的键盘。**远红外控制组**中有两个接收器，你可以用手持式发射器上的按钮“1”和“2”键来切换。所以在接口板中可以分配两个功能到手持式发射器的每个键。可以通过交替“1”和“2”键来在两种任务之间切换。当然，“1”和“2”键也可以作普通键来使用。

在红外输入的属性窗口中，你可以用其顶部的标签在 **IR 1**、**IR 2** 和 **IR 1+2** 之间切换。如果你选择 **IR 1**，手持式发射器按下相应的键且先前已经通过“1”键设置为任务 1，IR 输入模块就只返回一个 1。另一方面，如果选择 **IR 2**，手持式发射器必须已经通过“2”键设置为任务 2。

但是如果你选择了 **IR 1+2**，手持式发射器的设置无关。这种情况下，你也可以将 1)))和 2)))键用作输入。

在程序模块中，这种选择显示在手持式发射器符号的右下角，显示为白色的“1”或者“2”。但是，如果选择了 **IR 1+2**，程序模块中就不显示数了。



9 面板模块和面板：概览

在 ROBO Pro 软件中，你可以定义自己的面板，面板可以简化复杂模型的控制。面板显示在你的电脑屏幕上，且只能在线模式下工作。对于这一主题，见 [3.7 “在线和下载模式的差别”](#) 一节。

要创建一个面板，请在功能栏选择“Panel”：

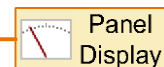


在空白的下面灰色区域，你可以插入面板模块。面板总是属于创建面板时所在的主程序或者子程序。因此，在你创建面板时，很重要的是你要在子程序栏选对正确的子程序，面板通常是在**主程序**下创建。

面板包含显示和控制模块。用显示模块，可以显示变量值或者文字信息。另一方面，用控制模块，如同附加的模拟量输入等传感器。



对于你插入面板的每一个面板模块，在程序中都有一个相应的模块。一个是“**面板输入(Panel input)**”（控制模块用），另一个是“**面板显示(Panel output)**”（显示模块用）。你可以通过这些程序模块在程序和面板之间



建立连接。你可以在“**输入，输出(Inputs, outputs)**”模块组找到它们。不同符号的显示，是根据你关联到程序模块的相应面板模块的不同。但是，在模块列表中只有两种模块：一个用于显示模块，另一个用于控制模块。

9.1 显示

“显示”模块的使用方法和“控制板输出”类似。可以用一个“=”指令传递一个数值到显示模块。

Cedutech®

9.1.1 仪表



仪表(Meter)模块基于一个带指针的模拟仪器。主要是用来显示模拟输入量。但是你也可以用它显示变量或者是其它的程序模块。

仪表模块是在程序中通过面板显示模块控制的。可以在“**输入, 输出(Panel output)**”模块组找到“**面板显示(Inputs, outputs)**”模块。

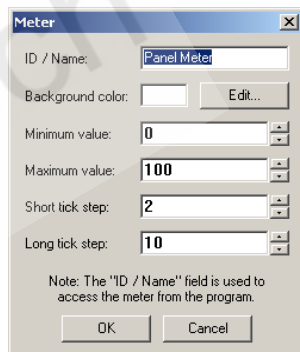


你可以通过一条“=”指令到程序中相应的“面板显示”模块来设定仪表值。几乎所有的带有数据输出的程序模块, 当这些值改变后都用“=”指令传递。你也可以直接将模拟量输入或者变量连接到“面板显示”模块。



仪表模块的属性窗口:

- 在“**ID/名称(Name)**”一栏, 你可以先输入仪表名字。输入名字很重要, 因为你可能要在程序中区别很多仪表。
- 在“**背景颜色(Background color)**”一栏, 除了白色你可以设置另外一种颜色。
- 在“**起始刻度值(Minimum value)**”和“**满量程刻度值(Maximum value)**”: 你可以定义刻度左右端点相应的数值。如果其中一个的值小于 0 而另一个大于 0, 那么指针初始就不会在最左端, 而是指向 0 刻度。
- 刻度有长短两种。在长短刻度间的距离可以在“**长线刻度/短线刻度(Step size short / long marks)**”一栏输入。如果是同样值, 仅仅有长线刻度显示。

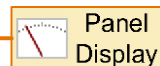


9.1.2 文本显示



在“文本显示(Text display)”模块中，可以用来显示数值，文本，或者是两者都有。

文本显示是在程序通过面板显示控制的。你可以在“输入、输出(Inputs,outputs)”模块组中找到“面板显示(Panel output)”模块。

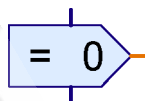


一旦你在面板输出的参数窗口中将它和文本显示相关联，这个符号改变了，且显示了文本显示的名称。

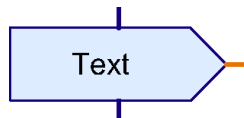


你可以通过两种方式，设置文本显示：

- 可以通过赋值“=”指令将显示的内容传递到程序中相应的“面板显示”模块。这非常实用，特别是如果要用它来显示变量或者是其它的程序模块，因为大部分的程序模块会在值改变时，自动通过它们的数据输出发出“=”指令。这条“=”指令仅仅改写显示模块的最后 6 个字符。你可以用一个预先填入的文本来填充余下的位置。用这种方式，你可以加上对所显示的值的一些解释性说明。如果是多行显示，你可以在某一行上加上解释文字。在多行显示时，仅有最后一行的最后 6 个字符通过“=”指令被改写。



- 你可以用文本指令设置你所要显示的内容。文本指令是个特殊的指令模块，通过它的输出不仅可以传递数据，还可以是完整的文本。和普通的指令模块一样，文本(Text)指令模块也可以带一个数据输入端。这种情况下，你也可以在文本中建立来自数据输入端的数值。如果你传递多个文本指令到一个显示模块，各个文本会连接起来。用此方法，你可以随意组合数据和文本。



文本指令中的控制字符：

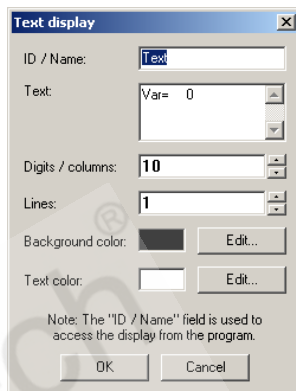
文本指令模块中可以使用以下的控制字符，来达到特殊的效果。

控制字符	效果
#####	将数据输入中数值以一个带“+”的 5 位数字符输出。
##.##	将数据输入中数值以一个带两位小数的数值输出，且用句号分隔。

##,##	将数据输入中数值以一个带两位小数的数值输出，且有逗号分隔。
\c	清除显示，并将后来的文本插入显示的开头。

文本显示的属性窗口：

- 在“ID/名字(Name)”一栏，先输入一个显示模块的名字。名字是很重要的，这样你可以在程序中的许多显示模块中把它区分开来。
- 在“文本(Text)”一栏，可以输入所要显示的内容。这些内容一直保留，直到你从程序中传递一个指令来改变它们。如果你传递一个“=”指令来显示，只有显示内容的最后 6 个字符被写出。正文的开头内保留，所以你在数值的前面可以显示一个注释来说明数值的种类。在图示的例子中，文本“Var=”保留。显示有 10 个字符，所以 10-6=4 个字符保留。
- 在“数字位数/列数(Digits/columns)”一栏和“行数(Lines)”一栏，可以设定显示空间的字符数。在一个多行显示中，你可以显示一个注释，象 Var=或者 Visitors 在自己一行中。
- 在“背景颜色(Background color)”一栏和“文字颜色(Text color)”一栏，可以改变显示的颜色设计。点击“编辑(Edit ...)”来选择一种颜色或者定义自己的颜色。



9.1.3 指示灯



“**指示灯(Display lamp)**”是最简单的一种显示类型。他的功能类似于连接到控制板输出的慧鱼小灯元件。

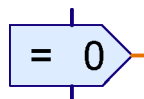
指示灯是用程序的面板输出来控制的。你会在输入和输出组中找到**面板输出(Panel output)**模块。



一旦你在面板输出的参数窗口中将它和指示灯相关联，这个符号改变了，且出现了指示灯的名称。

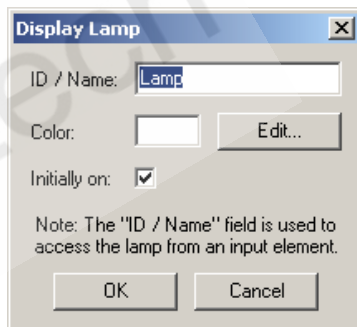


你可以通过传递一个相应的面板输出指令“**On**”和“**Off**”来打开或者关闭灯。正如你对真正的灯所做的那样。你也可以通过“**=**”指令来开闭灯。如果值大于 0，灯被打开。如果值小于等于 0，则灯被关闭。



指示灯的属性窗口:

- 在“**ID /名字(Name)**”一栏，你应该先输入一个指示灯的名字。名字是很重要的，这样你可以在程序中的许多指示灯中把它区分开来。
- 在“**颜色(Color)**”一栏，你可以改变指示灯的颜色。可以通过点击“**编辑(Edit)**”按钮实现。
- 如选中“**初始灯亮(Initially on)**”选项，指示灯一直点亮，直到相应的程序模块收到指令。否则指示灯初始时是关闭的。



9.2 控制模块

控制模块的用法和控制板输入是类似的。

Cedutech®

9.2.1 按钮

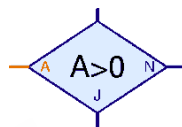
Button

你可以像一个连接到控制板输入端的慧鱼传感器或者开关来使用“按钮(Button)”面板模块。

按钮模块在程序中通过“面板输入(panel input)”模块查询。可以在输入输出模块组(Inputs,outputs)中找到“面板输入(Panel input)”模块。

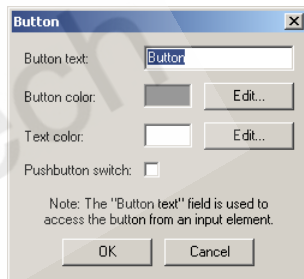


你还可以将按钮像控制板输入那样和面板输入关联，并输出到任何程序模块的数据输入端，比如判断模块。如果按钮被按下，返回“1”值，否则返回“0”值。



按钮的属性窗口：

- 在“按钮名称(Inscription text)”一栏，可以输入按钮的名字，这名字同时也是程序中访问按钮所用的。
- 可以在“按钮颜色(Button color)”一栏和“文字颜色(Text color)”一栏中改变按钮的颜色设计。并可以通过点击“编辑(Edit ...)”来实现。
- 如果选中了“按钮开关(Pressure switch)”一项，按钮作开关用而不是传感器。如果按一下按钮，那么它会保持压下状态直至第二次点击。否则，按钮就当传感器用，手松开压簧也直接打开了。

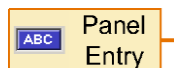


9.2.2 滑块

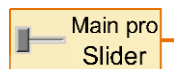


你可以像一个连接到控制板输入端的电位计一样来使用**滑块(Slider)**。和按钮只能返回“0”和“1”值不同，而是可以有许多不同的值。数值的范围可以通过属性窗口来设置。比如，滑块可以用来在 1 到 8 之间设定电机速度。

按钮模块在程序中通过“**面板输入(panel input)**”模块查询。可以在输入**输出模块组(Inputs,outputs)**中找到“**面板输入(Panel input)**”模块。



一旦你在面板输出的参数窗口中将它和滑块相关联，其符号改变了，且出现了滑块的名称。

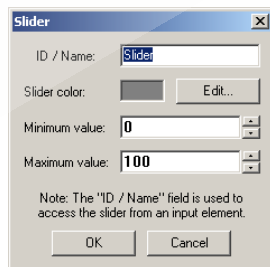


你可以将面板输出和滑块相关联，就像控制板的模拟输出，并输出到任何程序模块的数据输入端。滑块经常连接到一个带数据输入的指令模块。这样一来，就可以用滑块来控制电机的速度。



滑块的属性窗口：

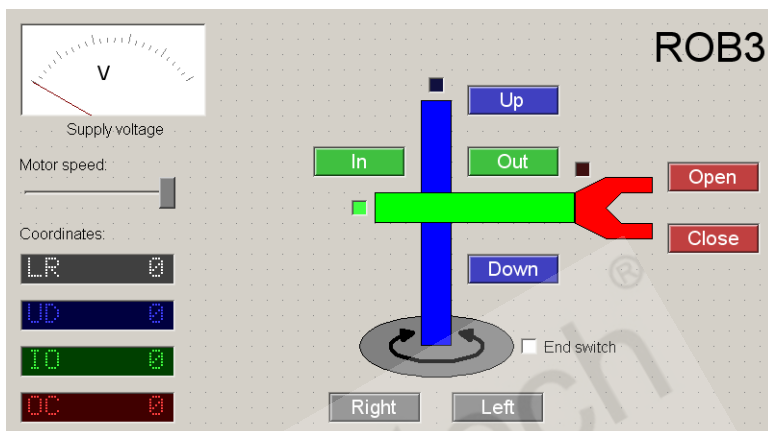
- 在“**ID/名字(Name)**”一栏，可以输入一个滑块名字。名字是很重要的，这样你可以在程序中的许多滑块中把它区分开来。
- 在“**滑块颜色(Slider color)**”一栏，可以改变滑块的颜色，点击“**编辑(Edit)**”就可以实现。
- 在“**最小值(Minimum value)**和**最大值(Maximum value)**”一栏，可以输入滑块的值的范围。如果你要用滑块控制电机的速度，值的范围应该是从 1 到 8。



ROBO Pro 还有普通的绘图功能。你可以在以下的**绘画模块组(Draw)**中学到它们的功能。子组“**形状(Shapes)**”包含用还绘制各种基本几何图形的工具。子组“**文字(Text)**”中。可以找到各种尺寸字体的书写工具。另一个子组包含了改变颜色和线粗细的功能。

10 绘图功能

用绘图功能，可以使你的面板和程序图可视化，使得它们的功能更清晰。下面的示例是一个用户设计的机器人。



按钮，坐标和终端开关指示灯在机器人的示意图中，对于各个单独的轴都保持相同的颜色。这样一来，各面板模块都很容易理解。

绘画功能的应用应该不会很难。所以只就以下不太清楚的几点内容作一下解释：

- 绘图对象，比如矩形，并不是像在其他许多程序中按下鼠标键画出来的，而是通过点击两次鼠标键，一次在矩形左上角，另一次在右下角。
- 文本并不是在对话框中，而是在工作区域中就可编辑。当插入一个新的文本对象，初始时只是显示一个明亮的蓝色框架。现在只须简单地用键盘输入，你输入的内容就直接显示在工作区域了。也可以用 **CTRL+V** 从剪贴板来插入文本。
- 一旦你建立了一个对象，你可以通过移动小蓝点来进行编辑。也有用来旋转和扭曲对象的手柄。矩形在其左上角有两个蓝点。如果你移动第二个大蓝点，可以移动矩形的角。点击鼠标右键或者按“**ESC**”键可以退出编辑模式。
- 如果要稍后再编辑对象，可以在“**绘制(Draw)**”菜单中选择“**编辑(Edit)**”功能。如果你点击对象，又出现了明亮的蓝点。

- 许多对象有两个以上的编辑和绘图模式。在绘制或者编辑对象时，可以用键盘上的“TAB”键在各个模式间切换。比如在绘制圆的时候，你可以选择定义两个边界点或者一个圆心、一个边界点等模式中选择。比如在编辑多边形时，可以在点编辑和“旋转”功能之间改变。对于文本对象，可以在编辑文本和改变字符大小或者旋转角度等功能间切换
- 在“**绘制(Draw)**”菜单中，有些功能可以“**将模块放到前层/后层(in the foreground / background)**”。用这个功能，你可以将选中的所有对象（红色绘制）放到前台或者后台，这样一来，使其它对象变暗。
- 用“**绘制(Draw)**”菜单中的“**模块栅格固定(Raster snap)**”功能，你可以打开或关闭字符矩阵。然而，你编辑程序的时候，务必注意到字符矩阵是打开的，因为所有的程序模块排列成了矩阵。
- 你可以通过按“CTRL”键加上数字键区的“1-9”中的一个，来改变文本对象的队列。但是这个功能只是在键盘上的“Num-Lock”灯点亮的时候才有效。如果其未亮，必须先按一下“NUM”键。

11 摄像头功能

4.x 或更高版本的 ROBO Pro 软件支持慧鱼 USB 摄像头，该摄像头可以通过 USB 数据线连接到 ROBOTICS TXT 控制板（通过 USB1 接口）。摄像头拍摄到的画面可以通过 USB 数据线或者 Wi-Fi 传输到电脑，并在 ROBO Pro 软件中显示。你也可以将摄像头作为一个探测颜色，运动，线条和小球的传感器，在在线模式和下载模式中均可以使用。

摄像头也可以直接通过 USB 数据线连接到电脑，这样 ROBO TX 控制板和 ROBO 接口板都可以在在线模式下识别摄像头信息。

Cedutech®

11.1 摄像头窗口

所有的摄像头设置均在摄像头窗口下(Camera)，如下所示：



在这个窗口下，首先选择**摄像头的连接方式(camera connection)**，是直接¹与电脑连接，还是与 TXT 控制板连接。勾选**激活摄像头(Active Camera)**选项，可以在窗口下显示摄像头拍摄的图像，该操作无需运行任何 ROBO Pro 程序。

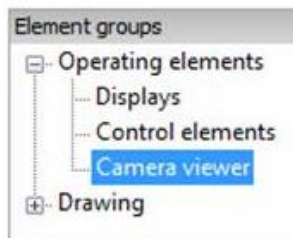
在摄像头窗口下，可以添加探测模块用于帮助机器人巡线。

当摄像头开启后，摄像头区域的信息可以显示在传感器数值(Sensor Values)区域，这样就可以知道当前有哪些传感器变量，具体数值是多少。

更多的详细信息将在接下来的章节中讲述。

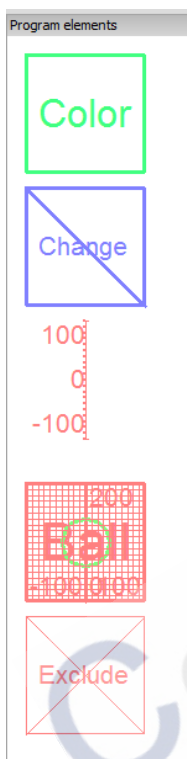
11.2 摄像头显示器

摄像头显示的图像除了在摄像头窗口中显示之外，也可以在**摄像头显示器(Camera viewer)**中显示。



在**面板模块(Operating elements)**下，包含有摄像头显示器。你可以在面板窗口中放置该模块，这样在线模式下，就可以看到摄像头拍摄的图像，就可以用来辅助遥控移动机器人。可以在**绘图(Draw)**菜单中的**编辑(Edit)**命令来更改显示器的尺寸。

11.3 摄像头探测模块



摄像头可以作为一个多功能传感器，ROBO Pro 软件提供了多种探测模块来满足这个功能。在摄像头窗口下，将模块栏中的探测模块拖拽到显示图像的画面中，并放置在合适的位置。

放置完成后，仍可以通过**绘图(Draw)**菜单下的**编辑(Edit)**命令来改变这些探测模块的尺寸。

右键单击这些模块，可以弹出模块的属性对话框，用以调整参数设置。

探测模块的数值可以通过橙色的**摄像头输入模块(Camera input element)**（级别3）读取，参与到程序控制中。



11.3.1 颜色探测器

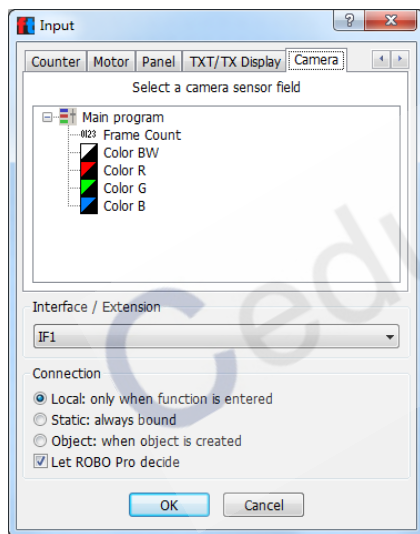


颜色探测器(Color detector)模块能够测算出矩形区域内的平均颜色。

你可以稍后通过绘图(Draw)菜单下的编辑(Edit)命令来改变这些探测模块的尺寸。

区域内所有的像素点都包括在内，确保能够测量到正确的平均值，即使是在复杂的图案区域，也会这样进行。

在属性窗口中输入模块的名称，然后就可以通过名称将该模块链接到输入模块。



颜色探测器功能提供 4 个信号数值，包括 **R,G,B** 和 **BW**，对应红，绿，蓝三原色，和亮度百分比（数值从 0 到 100，0 对应黑、暗，100 对应白、亮）。

请注意颜色数值会根据亮度变化。人类会把许多种光都看成无色透明，这些光可能由多种不同波段的光线组成，例如太阳光和电灯光，但是太阳光和电灯的光是不一样的。和人眼类似，摄像头也会尝试去补偿光的颜色，当摄像头探测的区域没有明显反光时，识别颜色比较准确。如果摄像头图像的全部区域都是彩色的，有可能颜色校准会有偏

差。

如果想要精确识别颜色，可以在稍后的线条探测器(line finder)中做到。

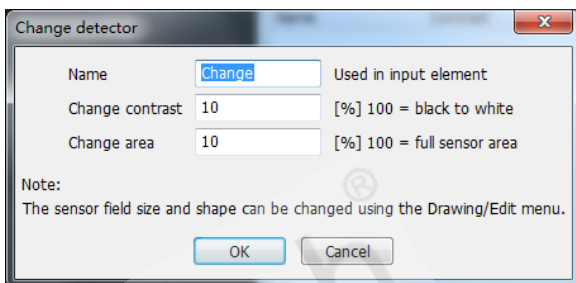
11.3.2 运动探测器



运动探测器(Movement detector)模块探测矩形区域内的图像是否有物体改变。你可以使用这个模块作为报警传感器，用于监测手势（例如挥手）或运动物体。

你可以稍后通过绘图(Draw)菜单下的编辑(Edit)命令来改变这些探测模块的尺寸。

- 在属性窗口中输入模块的名称，然后就可以通过名称将该模块链接到输入模块。



- 你需要在属性窗口中设置两个参数。**变化对比度(Change contrast)**指定像素点的亮度改变，100%对应从全黑与全白之间的变化。**变化区域(Change area)**指定矩形区域内变化的区域占的百分比。例如，如果想要探测白纸上蚂蚁的爬行，变化对比度设置比较高（50 或 50 以上），但是变化区域非常小（1）。

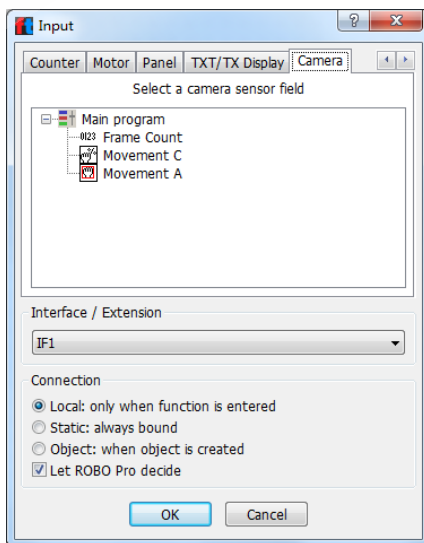


通过摄像头输入模块，可以看到运动探测器模块提供 2 个信号：C 和 A。

C 是对比度的平均值，像素点亮度改变的数值只有大于变化对比度设置中的数值，才会计入平均值。

A 指的是变化区域，是根据设定，变化对比度大于设定的那些像素点所占区域的百分比。

如果没有取得变化区域 A 的数值，变化对比度就为 0（没有变化）。因此，如果要探测运动，可以简单将 C 的数值与 0 作比较。

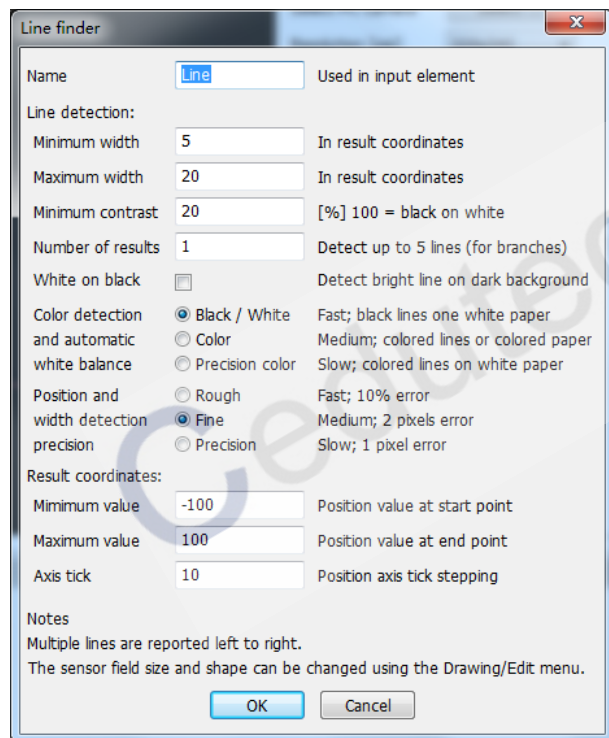


11.3.3 线条探测器



线条探测器模块(line finder)检测穿过模块刻度线的线条, 该模块能够检测线条的位置, 宽度和颜色。例如, 为了检测机器人应该沿着哪条线前进, 将该模块放置于图像上, 使其横穿整条线。

你可以稍后通过绘图(Draw)菜单下的编辑(Edit)命令来改变这些探测模块的尺寸。



在属性窗口中输入模块的名称, 然后就可以通过名称将该模块链接到输入模块。

在**最小宽度(Minimum width)**与**最大宽度(Maximum width)**栏中, 规定了给定识别区域线条宽度的范围。对于最大宽度, 设定要合适, 不能太大, 否则处理会非常耗时。同时, 对于最小宽度, 不能太小, 因为机器人快速移动中, 识别到的线条会变得模糊, 时而宽, 时而窄。对于宽度的设定, 要根据显示的刻度而定。最

好的办法就是, 在摄像头窗口下激活摄像头选项, 然后测量线条宽度, 然后减去或加上线条宽度 20%到 50%作为线条的最小宽度或最大宽度。

在**最小对比度(Minimum contrast)**一栏, 设置线条与背景的对比度。深黑线条在亮白背景上, 对比度为 100%。请注意, 机器人在移动过程中, 线条上某些点会因为反射, 使对比度下降。

在**线条数(Number of results)**一栏，可以设置要检测几条线。线条探测器功能模块能够检测到最多 5 条线，同时为每条线提供一系列独立信号输入给程序。线条会根据在刻度线上的位置输出，最靠近负刻度末端的线条先显示，也就是为线条 1，然后从负到正，为线条 2,3,4,5。

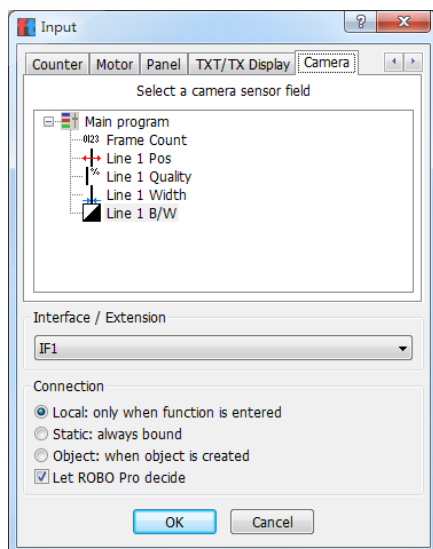
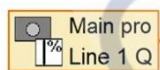
勾选**白线黑底(White on black)**选项，就会检测深色背景上的亮色线条。

颜色探测(Color detection)选项，可以允许你选择只检测黑白，或是包括彩色。在彩色模式下，红色线条在白色背景上的对比度与黑色线条在白色背景上相同。在彩色模式下，传感器模块同样提供了颜色的输入端，与颜色探测器(Color detector)功能相同。在精确颜色(Precision color)模式下，传感器模块使用白色背景，用来精确平衡白光。这种模式下检测到颜色更加精确，同时重复性较好，然而可能在某些情况下与颜色探测器检测到的颜色不一致。

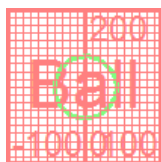
在**线条位置与宽度精度(Position and width detection precision)**选项中，可以设定传感器模块允许使用的计算时间，用以计算线条的位置和宽度。

在**最小与最大刻度(Minimum value/Maximum value)**栏中，可以设定刻度的范围，如果改变了这些刻度，需要调整线条宽度的设定。另外在刻度间距(Axis tick)中可以设置刻度之间的距离。

通过摄像头输入模块，可以看到线条探测器模块对每条线提供 4-7 个输入，这取决于上述的设置。最重要输入是**对比度(Line1Quality)**和**位置(Line1Pos)**。对比度输入能够提供线条与背景的对比度，如果对比度未能达到设定的阈值，返回值为 0，这样就能够确定是否检测到线条。**位置**是线条中心点在刻度上线的坐标。**线宽**是线条在刻度线上的宽度。另外，如果选择彩色模式，窗口中还会包含线条颜色 **R** 红色，**G** 绿色和 **B** 蓝色。

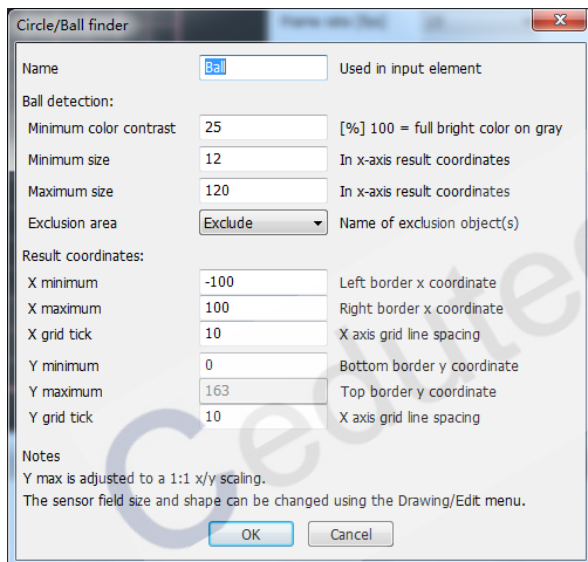


11.3.4 小球探测器



小球探测器(Ball finder)模块能够检测白色,灰色或黑色背景上的彩色圆形表面,小球或其他紧凑物体,同时能够给出物体的大小和位置。为了保证这个模块能够正常工作,只能检测彩色的物体。

你可以稍后通过绘图(Draw)菜单下的编辑(Edit)命令来改变这些探测模块的尺寸。



在属性窗口中输入模块的名称,然后就可以通过名称将该模块链接到输入模块。

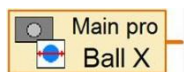
在**最小颜色对比度 (Minimum color contrast)**一栏,设置彩色物体与背景的最小对比度,100%对应高亮彩色在灰色背景上。

在**最小尺寸 (Minimum size)**和**最大尺寸 (Maximum size)**栏中,设置要检测物体的最小与最大

尺寸。对于最大尺寸,设定要合适,不能太大,否则处理会非常耗时,还可能会找到一些不像检测的物体。同时,对于最小尺寸,不能太小,因为在快速移动中,识别到的物体会变得模糊,时而大,时而小。对于尺寸的设定,要根据显示的刻度而定。最好的办法就是,在摄像头窗口下激活摄像头选项,然后测量物体尺寸,然后减去或加上尺寸的 20%作为最小尺寸和最大尺寸。

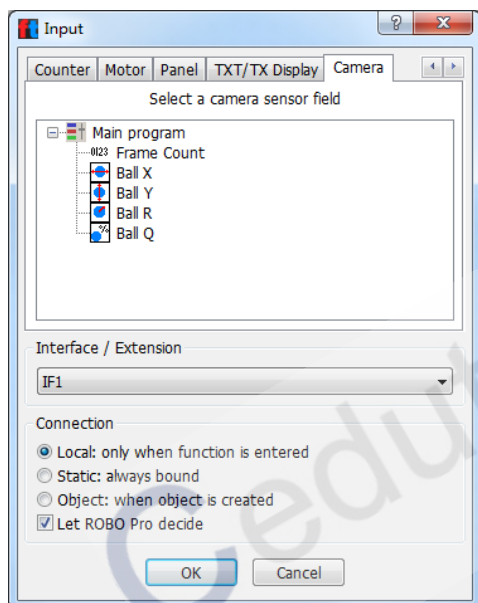
你可以选择是否在**排除区域(Exclusion area)**排除一些物体(详见下一节)。如果在检测区域有模型上的彩色零件,你可以将这些表面排除出检测区域。所有相同名称的排除模块使用时保持同步。

在**探测区坐标(Result coordinates)**中，可以设置检测区域刻度范围，Y 坐标会自动与 X 坐标保持一致，在**网格间距(X/Y grid ticks)**中，可以设置网格线的间距，即网格的疏密。



通过摄像头输入模块，可以看到小球探测器模块提供 4 个输入。最重要的输入是**对比度(Ball Q)**和**坐标位置(X/Y)**。对比度输入能够提供线条与背景的对比度，如果

对比度未能达到设定的阈值，返回值为 0，这样就能够确定是否检测到线条。**X/Y 坐标(Ball X/Y)**位置时被检测到物体的中心点。**尺寸(Ball R)**是被检测到物体在网格坐标下的半径。



11.3.5 排除物体



排除物体(Exclusion object)模块用于使探测区域中的某些部分不进行探测，与小球探测器模块配合使用。例如模型上的彩色零件可能被错误识别为小球，这时就需要排除物体模块。

你可以稍后通过绘图(Draw)菜单下的编辑(Edit)命令来改变这些探测模块的尺寸。

在属性窗口中输入模块的名称，然后就可以通过名称将该模块链接到输入模块。所有相同名称的排除模块使用时保持同步。

12 TXT 和 TX 控制板功能

ROBO Pro 软件 4.x 以上版本适用于 ROBOTICS 控制板和 ROBO TX 控制板，也适用于早期的 ROBO 接口板。编程要求没有变化。但由于控制板硬件上的差异，模型程序不能保持完全一致。ROBOTICS TXT 控制板的输入端与 ROBO TX 控制板完全一致，为了简明扼要，下面的内容基于 ROBO TXT 控制板。所有对于 ROBOTICS TXT 控制板的方法都可以应用于 ROBO TX 控制板。例如，ROBOTICS TXT 控制板有 8 路通用输入端口，可以同时输入数字量和模拟量，相反，ROBO 接口板仅有两路模拟量电阻输入端口（AX 和 AY）。同时，ROBO 接口板有电压测量端口，这些功能都已集成在 ROBOTICS TXT 控制板通用输入端口。

12.1 安装 ROBO TX 控制板 USB 驱动

在 ROBO Pro 安装文件中找到 USB-driver installation\TXController 文件夹，选择与你操作系统匹配的驱动。安装过程与 ROBO 接口板相似，参见 [1.2 安装 USB 驱动](#) 相关内容。

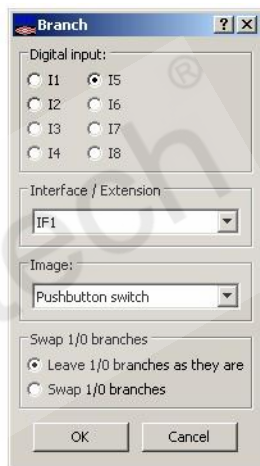
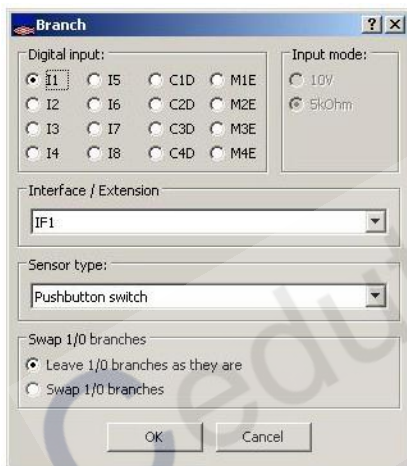
Cedutech®

12.2 编程环境（级别 1 及以上）

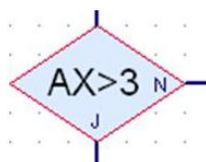
编程之前，你需要在工具栏选项选定编程环境，即连接控制板的类型为 ROBO 接口板或 ROBOTICS TXT 控制板。

选定好控制板类型后，工具栏将显示控制板图标，表明当前编程环境。这个按钮并不会改变当前的 ROBOPro 程序或者控制板的连接方式（连接方式只能通过 **COM/USB** 按钮设置）。

这个按钮会决定属性窗口中显示的内容。下图为 ROBO 接口板和 ROBOTICS TXT 控制板编程环境下分支模块。



此外不匹配的指令模块会用红色边框标记。大多数情况下，指令模块匹配。



12.3 控制板独立编程

只要你的程序只使用输入端口，命令适用于 **ROBOTICS TXT** 控制板和 **ROBO** 接口板，不需要在接口板和控制板上对程序进行改动。输入端口映射如下：

ROBO 接口板	ROBO TX 控制板
D1（超声波）	I1（超声波）
A1（模拟量 10V）	I2（模拟量 10V）
AX（模拟量 5kOhm）	I3（模拟量 5kOhm）
AY（模拟量 5kOhm）	I4（模拟量 5kOhm）
I1-I4（数字量）	I5-I8（数字量）
I5-I8（数字量）	C1D-C4D（数字量，不含轨迹传感器）

小贴士 1: **ROBOTICS TXT** 控制板连接的超声波传感器为 3 个接头，货号为：133009；**ROBO** 接口板连接的超声波传感器为 2 个接头，货号为：128597。

小贴士 2: **C1D** 简写表示 **C1** 端口作为数字量输入端，当作为快速计数输入端口时，简写成 **C1C**。

如果你的程序只使用上面列出的输入，通用输入端 **I1-I8** 的输入模式与 **ROBOTICS TXT** 控制板的输入模式匹配，直接可以将程序加载到 **ROBO** 接口板和 **ROBOTICS TXT** 控制板上。在在线或下载模式运行程序，程序能够自动完成端口识别。所以，你可以使用 **ROBO** 接口板的输入编写程序，但选择 **COM/USB** 中的连接方式连接到 **ROBOTICS TXT** 控制板。

12.4 程序转换

如果你不能或者不希望对接口板独立编程，你也可以将程序中的接口板编程环境设定为固定模式。以输入端口映射列表为参考，

Environment / Map inputs 菜单可以将所有输入转换到所选环境下的输入。在上表中没有列出的输入，如 D2, A2, AV, 不会自动转换，但可以随后手动转换。通过切换环境和再次调用菜单功能，您可以撤消此操作。

Cedutech®

12.5 通用输入，传感器类型和输入模式

在 ROBO 接口板中，每个输入端都有一个固定的输入类型。对 AX 输入端，只可以连接电阻传感器。ROBOTICS TXT 控制板，相比之下，有 8 个通用输入 I1-I8，可以由 ROBOPRO 程序控制，这样就可以连接不同类型的传感器。输入模式由传感器类型自动选择。在旧版本 ROBOPRO 中，可以对每个输入选择传感器图形，但这个只说明用途，没有技术功能。使用 ROBOTICS TXT 控制板，选择正确的传感器类型是很重要的。如果没有正确选择，输入就不能被正确识别。

在级别 4 或更高级别，你也可以在传感器类型中独立改变输入模式。

在 ROBOTICS TXT 控制板中，有些传感器尽管可以连接到 ROBO 接口板上的 I1-I8 接口，仍需要不同的输入模式。这主要对于轨迹传感器，在 TX 控制板中使用需要 10V 的数字输入模式。在程序转换与对接口板独立编程中为了选择正确的输入模式，ROBOPRO 使用之前的传感器图形作为传感器类型。

12.6 快速计数输入和扩展电机控制

ROBOTICS TXT 控制板有 4 路快速计数输入 C1-C4 和一个集成电机控制板，用于电机精细控制。扩展电机控制提供两个功能，包括运转特定距离后自动制动与两个电机的同步，电机控制系统要求电机 M1 的旋转编码器连接到快速计数输入 C1，其余以此类推。

使用**自动制动模式**时，当到达设定的目标脉冲数，控制系统自动制动电机，控制系统还可以计算电机的制动距离并提早制动，这样即使使用高速电机和高频旋转编码器也可以精确到达指定位置。

使用**电机同步模式**时，两个电机被控制到同样的转速。这对于慧鱼模型小车十分有用，其在直线前进时就是采用这种工作方式。如果一个电机转速变慢，电机控制系统自动减慢同步的另一个电机的转速。

您还可以结合这两个功能，使电机以同步转速运行，产生指定的脉冲数。

12.6.1 带编码器的电机（级别 1）

为了恰当控制拥有内置脉冲发生器（编码器）的电机，程序中新增了编程指令，在级别 1 或更高的编程环境下都能够使用。

使用这个编程指令，你可以使电机运转指定脉冲数，也可以使电机同步运转，指定脉冲数设定与否均可以。这个编程元素提供以下控制选项：

如果你想使一个电机运转指定的脉冲数，选择 **action distance**，设定 **speed**、**direction** 和 **distance**。

利用 **action synchronous**，你可以使两个电机同步转动，你可以独立设定两个电机的**转向**，由于两个电机转速相同，**转速**只需设置一次。

Action synchronous distance 集成了以上两种功能。

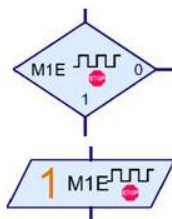
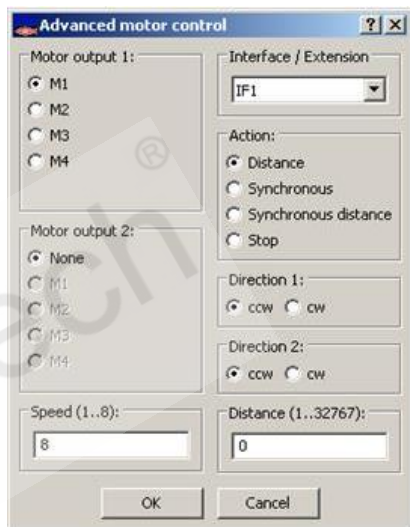
Action stop 中，你可以在任意时间停止电机，也可以在合适情况下结束同步和清除剩余距离。如果你使用此功能启动电机，你也必须使用此功能停止电机，之后才能使用电机的常用控制指令。

如果你设定指定距离，程序会直接进入下一条指令，**不会等待**设定距离到达。这样程序就会继续，在遇到特殊情况时可以自动停止电机运转。为了检测电机是否达到目标点，电机内置了 **M1E** 到 **M4E** 每个电机一个输入。你可以用一个分支或者一个**输入等待指令**来检测这些输入。



当相应的电机到达指定脉冲时，输入等待 **M1E-M4E** 变为 1，直到制定新的脉冲值，因此对于输入等待指令，最好等待如图中的数字量 1。如果你使用电机同步模式，只需要为一个电机设定输入等待。当两个电机都到达指定终点时，输入等待变为 1。

以上功能在 [4.4TANGO](#) 章节有详细介绍。

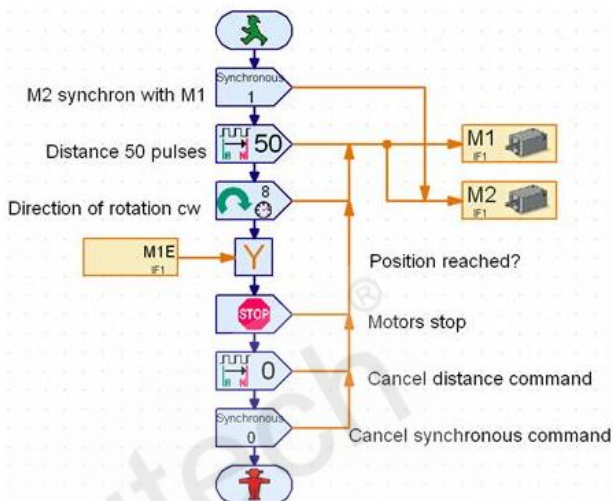


12.6.2 级别 3 中的扩展电机控制

在级别 3 中，电机由发送到橙色电机指令的命令控制。

使用 **synchronous** 命令，可以同步两个电机。例如，发送数值为 1 的同步指令到电机 2，电机 2 就与电机 1 同步。在级别 3 中，可以同步两个以上电机。如果发送同步指令值为 0，电机同步取消。

使用 **distance** 命令，你可以设定电机脉冲数，如果达到脉冲数，电机制动。指令值为 0 时，设定距离可以随时取消。



如果你组合使用同步和距离命令，两个电机都必须设定距离。而同步命令只需对一个电机设定，另一个电机的标号作为命令值。

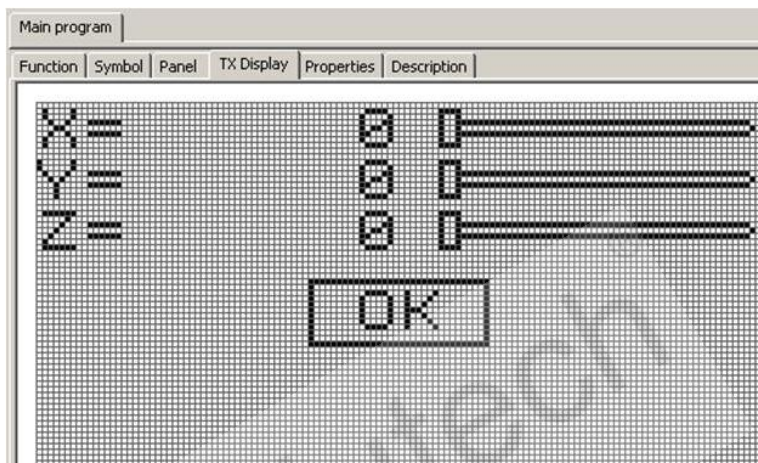
同步与距离命令都不起动电机，你需要选择转向命令。

你需要目标到达指令，这与级别 1 中的情况一样。当然，级别 3 中也有目标到达输入指令。

这之后，为了再次用普通电机指令控制电机，你必须首先发送值为 0 的距离和同步指令，以取消距离和同步功能。但是**严格意义上**，你需要向电机发出停止命令。距离和同步指令只会在命令使用时停止电机运转。如果取消距离和同步命令，没有制动电机，电机将继续运转。

12.7 显示屏

ROBOTICS TXT 控制板的另一个新特征是显示屏。类似于操作面板，显示屏用于控制程序或显示数据。**TXT/TX Display** 选项卡中的显示屏与操作面板中的一致：



可用的控制指令也与操作面板中的一致：有滑块和按钮。文本显示屏用于显示状态数据。为了结构化显示区域，提供了行元素和矩形元素。

小贴士：如果你想改变控制元素尺寸，可以使用 **Draw/Edit** 选项。

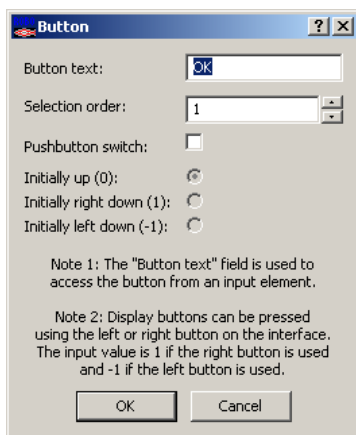
连接显示元素与程序方法与连接操作面板中输入输出元素方法相似。



显示功能通过接口板上的两个按钮控制，你可以通过短时间按左键或右键选择不同的控制对象。如果长按左键或右键，控制状态就改变了。

需要在属性窗口对每个控制单元选定一个序号，用按钮对其进行选定。

小贴士：如果你在使用显示屏功能的下载模式下停止程序，必须同时按下两个按钮。

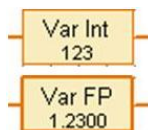


与操作面板相似，每个子程序也包含显示能。但是有些差异：显示内容会随着子程序的进入与退出自动改变。这种方式可以毫不费力地开发相当复杂的菜单结构。这里建议用简单的过程运行含有显示内容的子程序，否则在何种情况下显示何种内容将会变得难以预料。

Cedutech®

13 小数功能

ROBO Pro 提供小数的功能（也称为十进制小数）。意味着在数字运算时，不仅可以用 1, 2 这样的整数，还可以用 3.99482925 这样拥有 9 位有效数字的小数。在在线和下载模式下，ROBOTICS TXT 控制板和 ROBO TX 控制板通过浮点计算功能实现以上运算。ROBO 接口板只能在线模式下运行浮点运算。

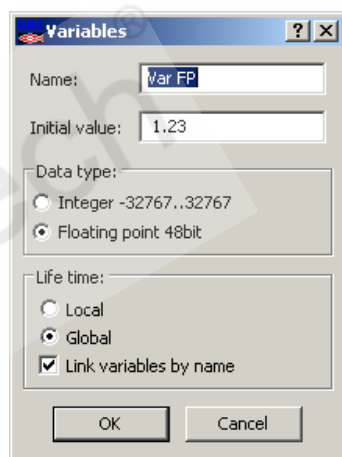


如果你注意细节：运算精度是拥有 32 位小数部分的 48 位二进制数字，这略高于十进制的 9 位有效数字的精度。

ROBO Pro2.1.1.0 版本支持以下功能：

- 浮点变量
- 浮点列表
- 加减乘除运算
- 整数浮点数相互转换
- 浮点数和常量的比较指令
- 具有浮点数的文本命令

软件中没有提供专门的浮点元素，需要在数据属性窗口设定数据类型，浮点数以黑体显示。



13.1 对比浮点数

浮点数不能简单地比大小，因为浮点数不会相等，这是由于浮点数的数值由于进位而不准确。例如，10 个 0.1 之和不是 1，由于 0.1 不能用二进制浮点数准确表示。

你可以在级别 3 中的比较模块比较浮点数与浮点常量的大小，同样还有浮点数比较运算功能。

Cedutech®

13.2 显示浮点数

相比于电脑显示器, ROBO TX 控制板显示屏大小有限, 所以 ROBO Pro 软件提供了节省办法显示浮点数。指数用工程中常用的指数表示法表示, 例如 km 中的“k”表示“千”。指数的缩写列表如下:

缩写	名称	指数
a	atto	10^{-18}
f	femto	10^{-15}
p	pico	10^{-12}
n	nano	10^{-9}
u	mikro	10^{-6}
m	milli	10^{-3}
k	kilo	10^3
M	Mega	10^6
G	Giga	10^9
T	Tera	10^{12}
P	Peta	10^{15}
E	Exa	10^{18}

为了防止指数超出范围, 计算中常见的计算错误都被显示出。

当然, 浮点数也可以用不同于电脑与计算器中的表示方法, 文本命令提供以下功能:

格式	输出 1	输出-0.01	输出 1000
####.####	__1.0000	__ -0.0100	?FORMAT?
#####	__1	__ -0	_1000
##.###^	_1.000	-10.00m	_1.000k
##.###^##	_1.000^00	_1.000^02	_1.000^03
##.#####^#####	_1.0000E+0000	_1.0000E-0002	_1.0000E+0003

例如:

两个常数相加, 结果 5 秒后显示。然后显示被清除 (文本命令中输入\C), 之后“End”字符显示。

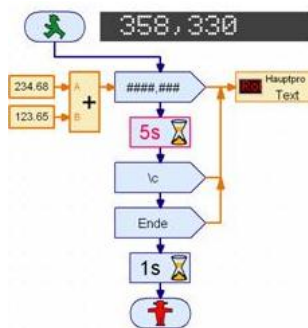
请注意以下关于格式的提示:

- 有效数字的位数及指数都可以通过所有方式表示;
- 您可以使用一个点或逗号作为小数点分隔符;

- 点或逗号前至少有 2 个 # 字符, 1 个表示符号, 1 个用于小数点分隔符前面的至少 1 位数字。

ROBO Pro 使用下面的代码显示特殊值和标记错误信息:

- **0** 是用来表示一个确切的零 (没有错误) 或约小于 $\pm 10^{-2500}$ 的数字;
- **?FORMAT?** 数量不能使用所选的格式显示;
- **?OVERFLOW?** 计算结果算术溢出。例如, 除以零结果溢出;
- **?NAN?** 没有一个数字是的结果, 无效计算, 如 -1 的平方根;
- **?UNDEFINED?** 例如, 这个值可以用于子程序输入在接收到数值之前的显示值;
- **?LOST?** 用于显示输入 0/0 等;
- **?CORRUPTED?** 这不应该发生。如果你的程序显示这个值, 请致电慧鱼服务部门;
- **??.??** 见下一节。



13.3 计算精度

与其他大多数浮点系统不同，ROBO Pro 软件在每个操作中都进行计算，同样包括有效数字或位的位数。在计算过程中丢失的数字在文本中显示的输出为“?”。例如，计算 $1.00000001 - 1.00000000$ 的结果为 $9.8??n$ 。不同之处在于 0.00000001 与 $10n$ 的区别，然而，ROBO Pro 的显示数字比计算更加精确。最后一个数字仅仅是帮助确定四舍五入，如从 $9.8n$ 到 $10n$ 。如果你在 ROBOPro 计算 $1.0-1.0$ 结果是 $???P$ ，这意味着 0 有 $99.99p$ 或 $100P$ 的精度，就是有 $\pm 0.01p$ 的浮动。如前所述，存在一个确切的 0 （没有错误），但是这是不常见。

Cedutech®