圆梦小车 Step by Step 之二

—— 和小车“交流”

信息时代，交流乃第一需求，人如此，机器也不例外，它们也需要有交流的通道、交流的方式，以使它们的功能发挥到极致，同时也让它们对人更加友善。

传统设计中，小车与人的交流无一例外的选用了 LCD、LED 显示器，但我仔细分析后发现，使用这些显示器有许多缺憾，其一是显示内容有限，无法满足人的需要，也无法适应小车进化新增信息的显示；其二是小车是移动的，装在车上的显示器在需要观察信息时反而看不见了，因为它“逃出了”你的视野。

所以，圆梦小车的设计放弃了显示器，取而代之的是设计了一个 USB 转 UART 接口和一个无线接口，将需要显示的信息传递出来，借用已随处可见的 PC 机显示，无疑这样内容将会丰富、生动许多。还提供了无线接口，使小车在运动时也可将其需要表达的告诉你，使你随时知道它在干什么？是否有什么“不爽”的事发生？

由于通讯是双向的，所以还可以取代原来小车上的控制按键，圆梦小车上除了一个电源开关外，无一按键。通讯实现的控制远比按键灵活，将会给你无穷的发挥空间。

一、 交流的基础

1.1 硬件

首先，交流需要一定的硬件支持，就像人必须有“嘴”和“耳”一样。

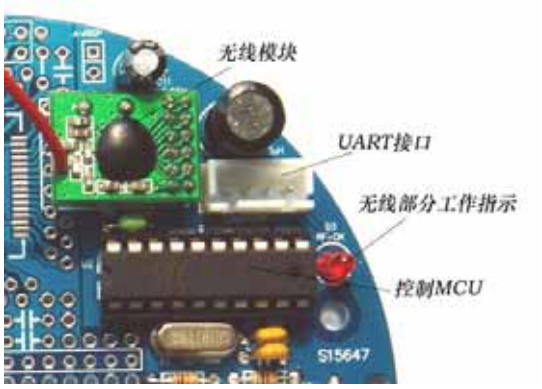
串口是伴随计算机而生、资格最老的通讯接口，而且也是最成熟的，由于其简单易用性在单片机中广为采用。但是 PC 机发展到今天，已基本将其淘汰，特别是笔记本电脑上已难觅其踪影，取而代之的是 USB 接口。

所幸，器件厂家为了兼顾不同领域的需求，推出了 USB 转串口芯片，用 USB 口实现串口的功能，其仿真度还相当不错，在 PC 机上编程时，完全可以当成一个标准的串口使用，唯一的麻烦就是需要安装一个驱动程序。

所以，在圆梦小车设计时，将 USB 转串口作为基本配置，使小车可以直接和 PC 相连。

USB 连接在初期调试时问题不大，小车是要运动的，此时它就成了拖后腿的“ 尾巴 ”了。所以设计一个无线接口是我期盼已久的！可无奈对于学习目的而言，性价比合适的无线模块难找啊！

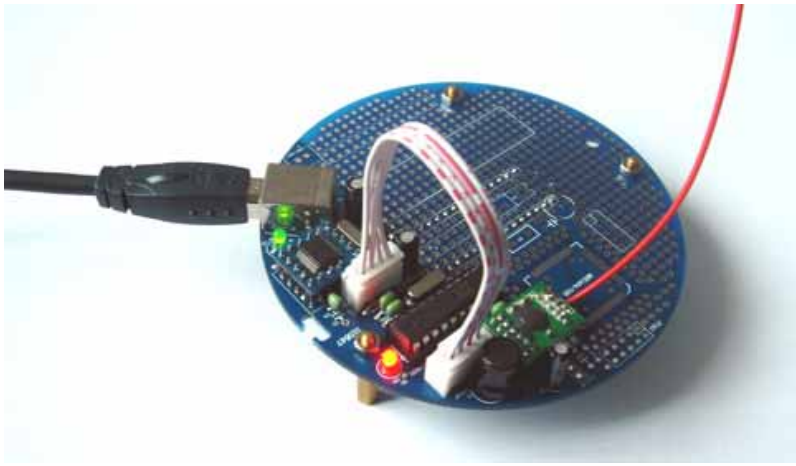
一个巧合发现了这款无线收发模块，20 多元的价位基本符合要求，经过一番试用，觉得性能还不错，就是控制它对于初学者而言略有难度，所以单独使用了一个 MCU 控制，将串口收到的数据交给无线模块转发出去，同时将无线模块收到的信息从串口送走。对于小车或其它串口设备而言，和有线通讯没有差别，也就是常说的“ 透明 ”方式。这个无线接口作为选件提供给大家。

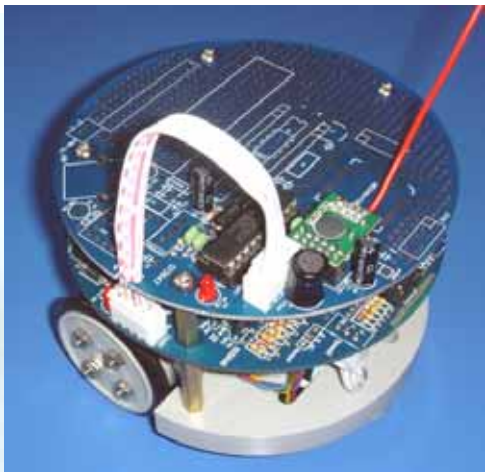


如需使用所提供的套件实现 PC 机与小车的无线通讯，建议按如下配置：

基本配置 + 2 套无线套件 + 1 块扩展 PCB

在基本配置中的扩展 PCB 上只焊无线接口部分，而在追加的扩展 PCB 上焊上 USB 转串口以及无线接口部分，见下图：





这样就等于给 PC 机增加了一个无线适配器，可以和小车无线通讯了。

1.2 通讯协议

人交流需要约定语言、语法、词义，机器之间也是同样，只是被称之为“通讯协议”。

- 首先约定“**语言**”—— 字节格式，此处为：

标准 UART 格式 —— **19200 8 N 1**

（波特率 19200 8 位数据位，无奇偶校验位 1 位停止位)

- 其次约定“**语法**”—— 数据帧格式，此处为：

帧头（2 字节） 接收方地址（1 字节） 发送方地址（1 字节） 帧长（1 字节）

命令（1 字节） 数据域（N 字节） 校验和（1 字节）

帧头 —— 由 2 个特殊的字节 **0x55 0xAA** 构成；人说话前要“哎”一声，提醒听者注

意，机器也是一样，需要一组特殊的信息提示接收方：“ 后面是有用的了 ”。

接收方地址 —— 通讯对象的 “ 名字 ”，在有线通讯时也许多余，但无线时就需要了。

发送方地址 —— 告诉接收方，是谁和你 “ 说话 ”，便于接收方回答。

帧长 —— 告知接收方，有多少字节信息，好让接收方知道什么时候 “ 完 ”；也有采用特殊的字符作为结束通知的，但那样会限制发送的内容，同时也让接收方 “ 心里没底 ”。所以此处采用了提前告知的方式，接收方可以根据长度确定自己是否能收下来？如果内存不够，可否采用变通的方式处理，如：边收边解析，或者只存有用的部分。

因为单片机内存有限，所以帧长使用 **1** 字节就够了。

命令 —— 是协议的核心，所有传递的内容由命令决定，它是表达 “ 意思 ” 的，下面在 “ 词义 ” 部分讨论它的细节。因为内容有限，所以只用了一个字节，可以表达 255 种意思。

数据域 —— 与命令配合，表达一个完整的含义。

校验和 —— 因为是串行通讯，在一连串数据中出现错误的可能较大，所以设置了检验和，将数据帧中从命令开始到数据域结束所有字节的算术和，取最低字节的反码。

可靠性要求高一些的场合一般使用 2 字节 CRC 校验码，但对于单片机来说运算有些困难，且不利于初学，所以没有采用。有兴趣者可以自己尝试。

➢ 最后定义 “ **词义** ” —— 命令定义：

命令的定义完全根据需求，通常是逐渐增加、完善。

因为还没有涉及任何外设，所以先定义两个与内存相关的命令 —— 读、写内存。

✧ **命令一 ：读内存**，实现读指定地址开始的 N 个字节，地址用两字节表示。

命令字 —— 0x01

数据域 —— 低地址（1 字节） 高地址（1 字节） 读字节数（1 字节）

地址与硬件的对应关系：

0x0000 — 0x00FF —— 对应 STC12LE5412 的 256 字节内部 RAM（idata）；

0x0100 — 0x01FF —— 对应 STC12LE5412 的 256 字节外部 RAM（xdata）；

0x0200 — 0x7FFF —— 保留，为大 RAM 的单片机预留；

0x7F80 — 0x7FFF —— 对应 STC12LE5412 的 128 字节 SFR；

0x8000 — 0xAFFF —— 对应 STC12LE5412 的 12K FlashROM（Code）；

0xB000 — 0xFFFF —— 保留，为大 ROM 的单片机预留；

例：要读地址 0x56 起始的 3 字节内部 RAM 数据，命令帧如下：

0x55 0xAA XX XX 0x04 0x01 0x56 0x00 0x03 0xA5

返回数据帧为:

帧头 发送方地址 自己的地址 帧长 命令 低地址 高地址 读字节数

N 字节数据 校验和

返回帧中将命令及附属信息（地址、读字节数 ）包含在内，虽然有些冗余，但保证了信息的完备性，不需要接收时还要查找原来读的是什么？为通讯需求日渐复杂提供方便。

✧ **命令二：写内存**，实现写指定地址开始的 N 个字节，地址用两字节表示。

命令字 —— 0x02

数据域 —— 低地址（1 字节） 高地址（1 字节） 写字节数（1 字节）数据（N 字节）

其地址与硬件的关系与读命令相同。

返回数据帧为：

帧头 发送方地址 自己的地址 帧长 命令 低地址 高地址 写成功字节数 校验和

通过“**写成功字节数**”来告之发送方是否写成功，如果为“0”，表示写操作失败。

命令暂且定义到此，后面根据需要逐渐扩充之。

二、 交流的实现

虽然通讯不一定局限于 PC 机与小车之间，也可以是小车和小车之间，但开始的需求多以 PC 机与小车的通讯为主，所以此处讨论的是 PC 机与小车的通讯。

2．1 小车侧的通讯实现

有了硬件基础，也有了协议，下面就要构思收、发的算法，编写通讯程序了。

STC12LE5412AD 是 51 系列单片机，其 UART 与标准 51 相同，而且只能使用 T0 或 T1 作为波特率发生器，这点有些落后于一些新的 51 单片机，它们多有独立的波特率发生器，可以腾出一个定时器留作它用；好在此 MCU 增加了 4 路 PCA（可编程计数器阵列）弥补了这个不足。

不过还是有一点改进，因为 STC12LE5412AD 的定时器可以选择用 12 分频计时还是 1 分频计时，所以使用同样晶振时其波特率可高于标准 51 单片机 12 倍。

对于通讯处理程序，难在“ 接收 ”而非“ 发送 ”，因为“ 发送 ”是主动行为，按需构建好数据帧后，放入发送缓冲区，由发送程序一个个送出去即可。

可“ 接收 ”是被动行为，不知道何时会来？也不知道什么时候算是有效信息？什么时候是“ 噪声 ”？什么时候算是说完了？听错没有？这一切都需要通过程序来甄别，所以畅通、可靠的交流主要取决于接收程序的好坏！

先易后难，将发送处理构建好再说！

发送需要以下几个变量：

ga_ucTxdBuf[MaxTxdByte_C] —— 发送缓冲区，存放将要发送的数据帧；

gi_ucTxdPtr —— 发送指针，发送缓冲数组的下标，指向将要发送的字节；

gc_ucTxdCnt —— 发送字节计数，构建发送帧后赋初值，之后减计数到“ 0 ”完成。

由上面三个变量即可看出发送的操作了：

先按需要将发送帧组织到发送缓冲区，将发送指针指向第一个字节，发送字节数为帧长加 10，将第一字节送入 SBUF，即启动了发送；之后在串口的发送中断处理中先检测发送字节计数，不为零则发送字节减计数，发送指针加 1，取数送 SBUF。

详细看所附程序。

现在来解决难题 —— 接收处理！

由于各种原因，一个接收器往往能不断收到讯息，如：处于通讯总线上或无线网络中，即使是点对点有线传输，也不排除由干扰造成的无效信息。所以接收程序必须具备：在信息序列中摘出有效的部分，而不能奢望收到的第一个字节为有效数据。设置帧头的目的就在于此 —— 让接收程序便于找到有效的信息。

因为接收方无法知道何时开始是有效的数据，那只有不断的接收，缓冲区是有限的，如何才能实现连续不断的接收呢？

环形缓冲区是一个很好的解决办法，将接收缓冲区构建为一个头尾相连的数据环，则将有限的缓冲区等效为无限了，只要有效数据帧长度小于环形缓冲区，则该数据帧就不会丢失。

处理程序需要的变量有：

ga_ucRcvBuf[MaxRcvByte_C] —— 接收缓冲区；

gi_ucSavePtr —— 将收到的数据存放到接收缓冲区的指针，即数组下标；

gi_ucGetPtr —— 从接收缓冲区中取出数据的指针；

gi_ucStartPtr —— 收到有效帧头、帧长、接收方地址后，记下的命令字所处的位置；

gi_ucEndPtr —— 根据帧长计算出的帧结束位置。

从上述变量还不能看出环形缓冲区的概念，关键是缓冲区大小的设置及指针加减的处

理。

接收缓冲区的大小 MaxRcvByte_C 要设置为 2 的倍数，即 16、32、64、128，因为 16 以下太小，而 128 以上对于单片机来说有些不实际。

存数指针在处理时，每收到一个字节加 1，但是加后做一个“与”运算，即：

```
gi_ucSavePtr  =  (gi_ucSavePtr +1) & (MaxRcvByte_C -1)
```

读者可以自己分析一下这样处理的 gi_ucSavePtr 数值范围是什么？

同样，对于取数指针 gi_ucGetPtr 也是如此处理，只是取数指针还存在减运算，即：

```
gi_ucGetPtr = (gi_ucGetPtr +1) & (MaxRcvByte_C -1);
gi_ucGetPtr = (gi_ucGetPtr -1) & (MaxRcvByte_C -1);
```

通过这样的处理，接收缓冲区就成了一个没头没尾的环！这是接收处理的关键。其它细节请看程序。

至此，小车已经可以正确的发送或接收一个数据帧了。接下来的任务是对收到的数据帧解析，得到命令字，并根据命令字取出相应的参数，做相应的处理，这一切就不用文字叙述了，读者可在程序中看到。

2．2 PC 机侧的通讯实现

我一直是搞单片机的，以前有条件让别人为我做一些 PC 机上的辅助测试程序，就没有自己去尝试编写，但那时就觉得 PC 机真是一个很好的“帮手”。

现在没人可以随意差遣了，只好自己动手。通过一番广泛求助，总算能够借助 VC 编写最基本的程序了，我的定位很明确 —— 能够用 PC 机编写辅助单片机调试的工具程序，对 PC 机程序本身不做深入研究，力求用最简单的方式实现。我觉得，所有搞单片机的都应该掌握到此程度，因为 PC 机能给你带来太多的方便。

从我接触的朋友看，能做到此的不多，大多数人都是由于“畏惧”心态而放弃的。我自己入门的感受也是如此，开始觉得用 VC 编写一个在 Windows 下运行的程序很难，无从入

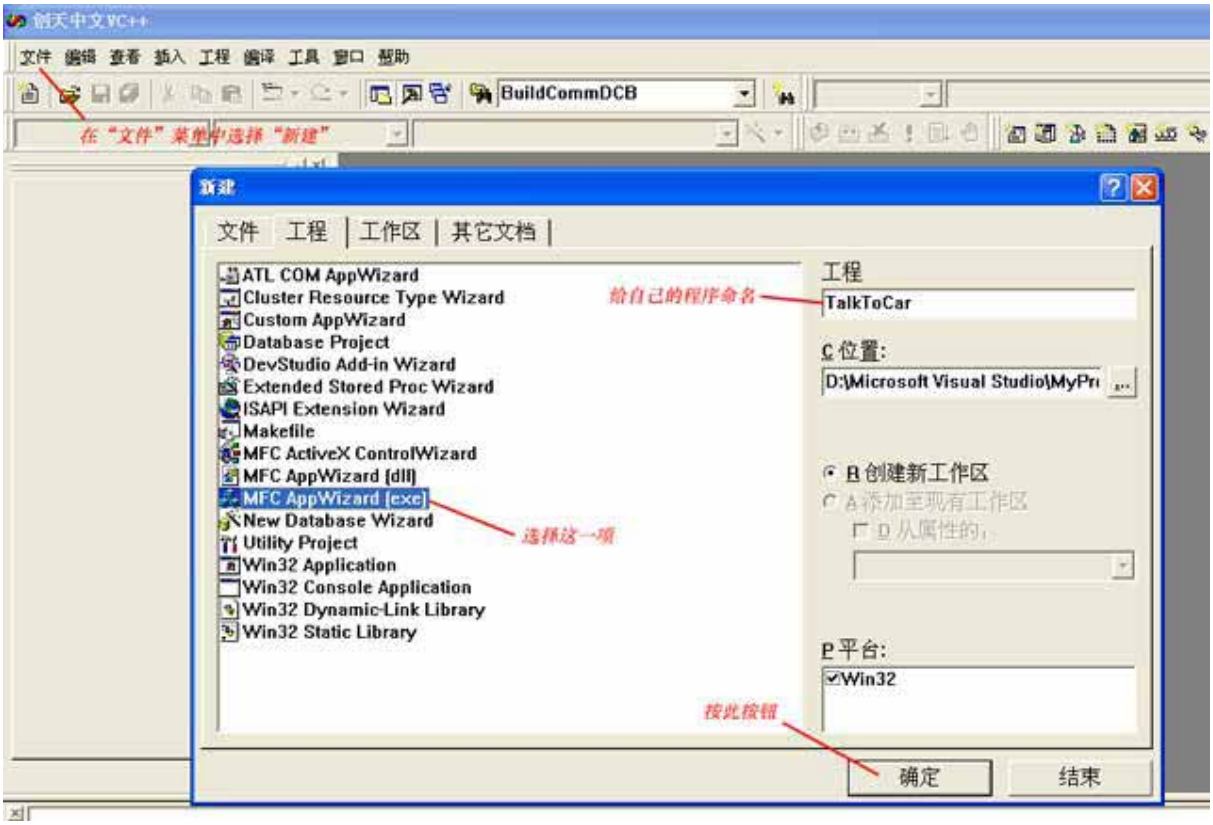
手，后来在一个高手指点下，很快上手，之后在各种应用中学会了不少新的内容，对 VC 也从“恐惧”、“排斥”变为喜爱了。

在此，我想尝试一下，用简单的、不正规的方式描述一下如何快速上手，期望对有此愿望的读者有所帮助，只需要你有基本的 C 语言编程基础。

由于我对 API 不了解，所以直接就用 MFC 了，反正都不知道，用号称包装的更好的 MFC 也许容易一些。

以编写一个与小车通讯的程序为例，**实现对指定地址一个字节的读、写**。

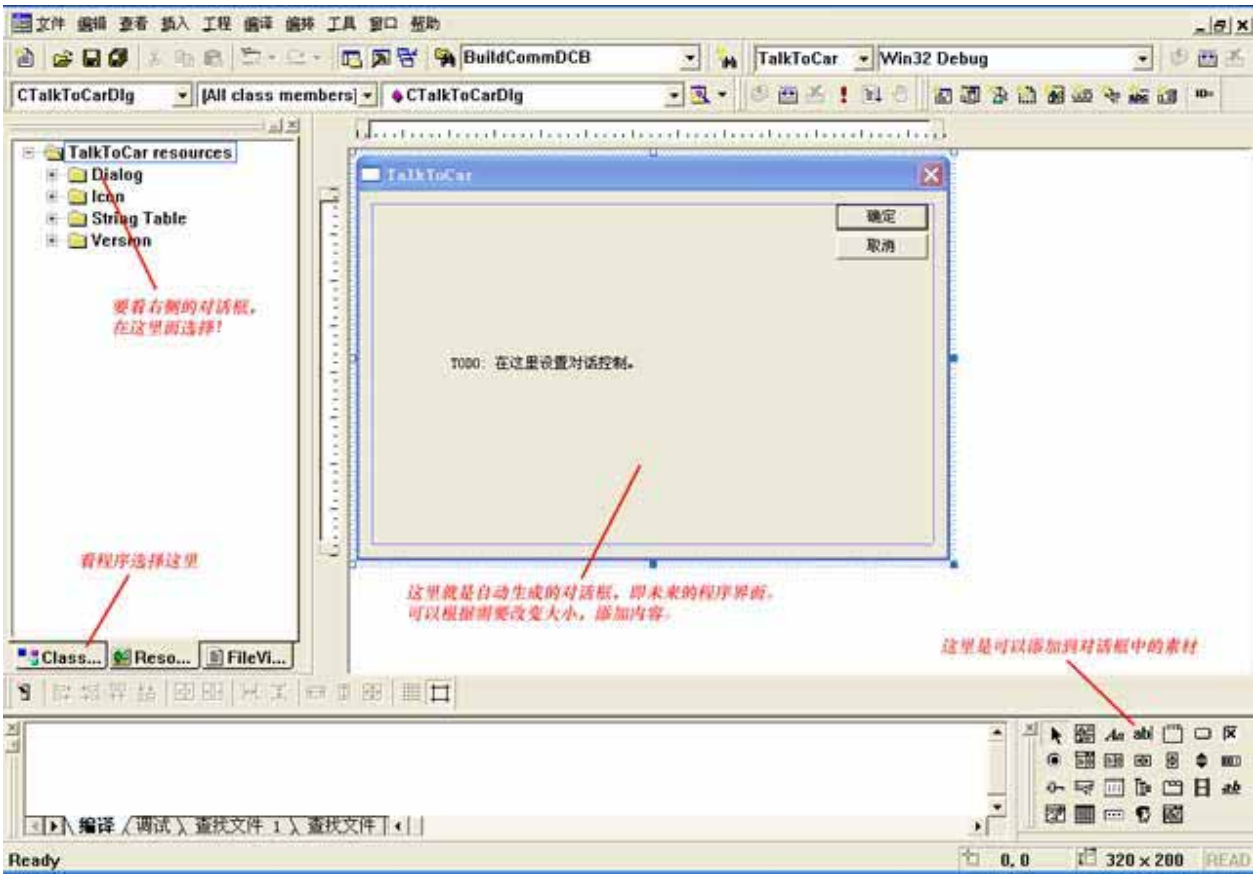
安装 VC++程序，运行后，第一步 —— 新建一个 MFC 应用程序：



第二步 —— 选择应用程序类型，我只会“基本对话”，也觉得够用了。直接选择“完成”，如果你比较“牛”，可以用“下一步”定制自己所需的内容。

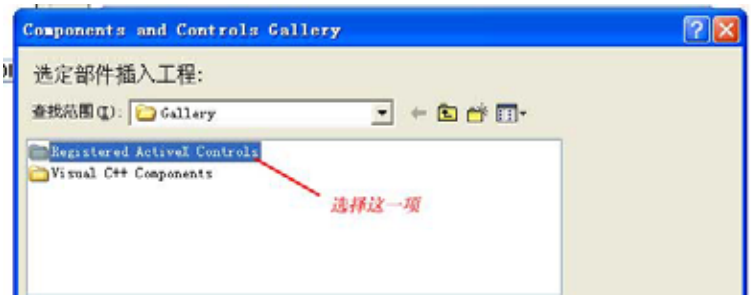
之后会显示如下窗口，告知你所建立的工程信息，有用的只是目录，否则找不到了。

确定后即进入真正的编程阶段，下面显示的是 VC 编程环境的主界面，左侧一栏可切换类（即源程序）和资源（显示界面等）查看，右侧为内容显示窗口，右下角一小块区域非常重要，是构建程序功能和显示的主要素材来源（资源）。

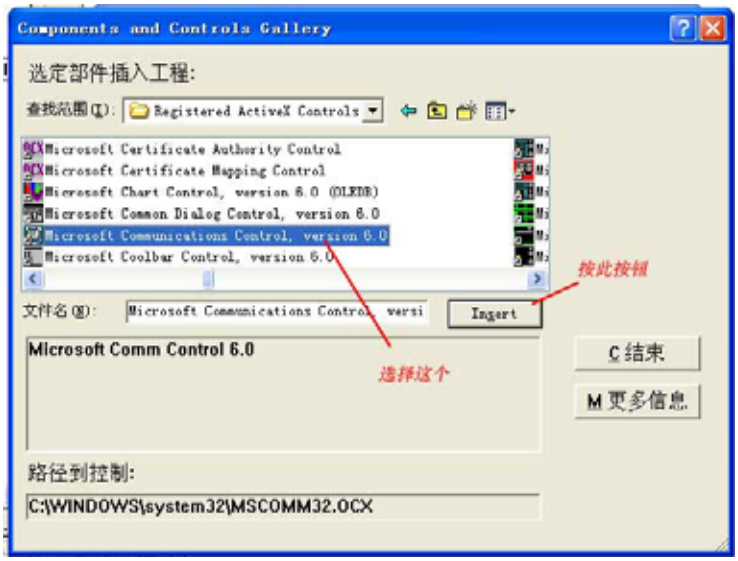


因为我们的程序需要使用串口通讯，所以先要进行如下操作，给它添加一个素材：

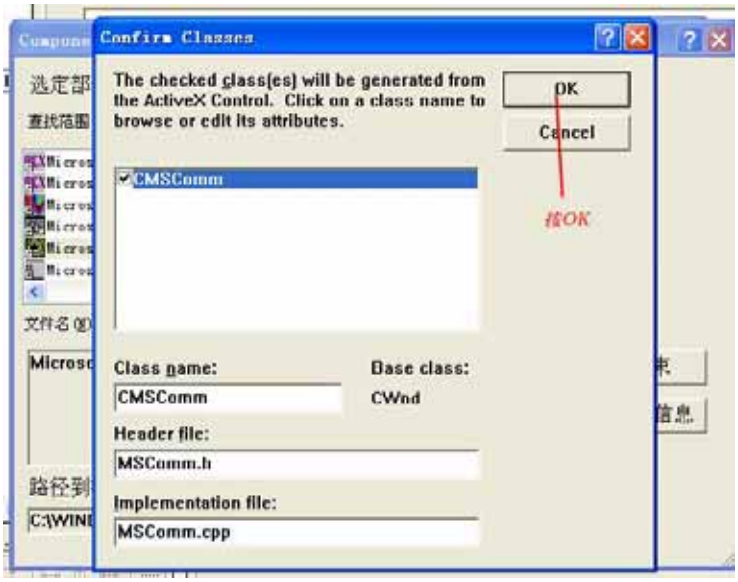
点击“ Components and Controls… ”后显示：



双击上面的选择，显示：



按字母顺序找到“ Microsoft Communications Control,V6.0 ”选择，并按“ Insert ”：



按“ OK ”后回到上一个界面，按“ 结束 ”即完成了通讯用控件的添加，注意添加后的素材区域增加了什么？

VC 编程与以前的概念有所不同，是先根据需要构建程序的显示界面，然后再将界面中的各个元素定义好，如按钮、显示内容栏等，在这个过程中 VC 帮你自动生成了对应的程序框架，并在框架中提示出添加处理功能的位置。

根据约定的功能：在小车内存中指定地址，读、写一字节。

首先要能选择串口，然后要能输入地址，显示读回的数据，还要能输入待写入的数据，此外，需要两个操作按钮，一个启动“读”，一个启动“写”。

按此构思设计的界面如下：

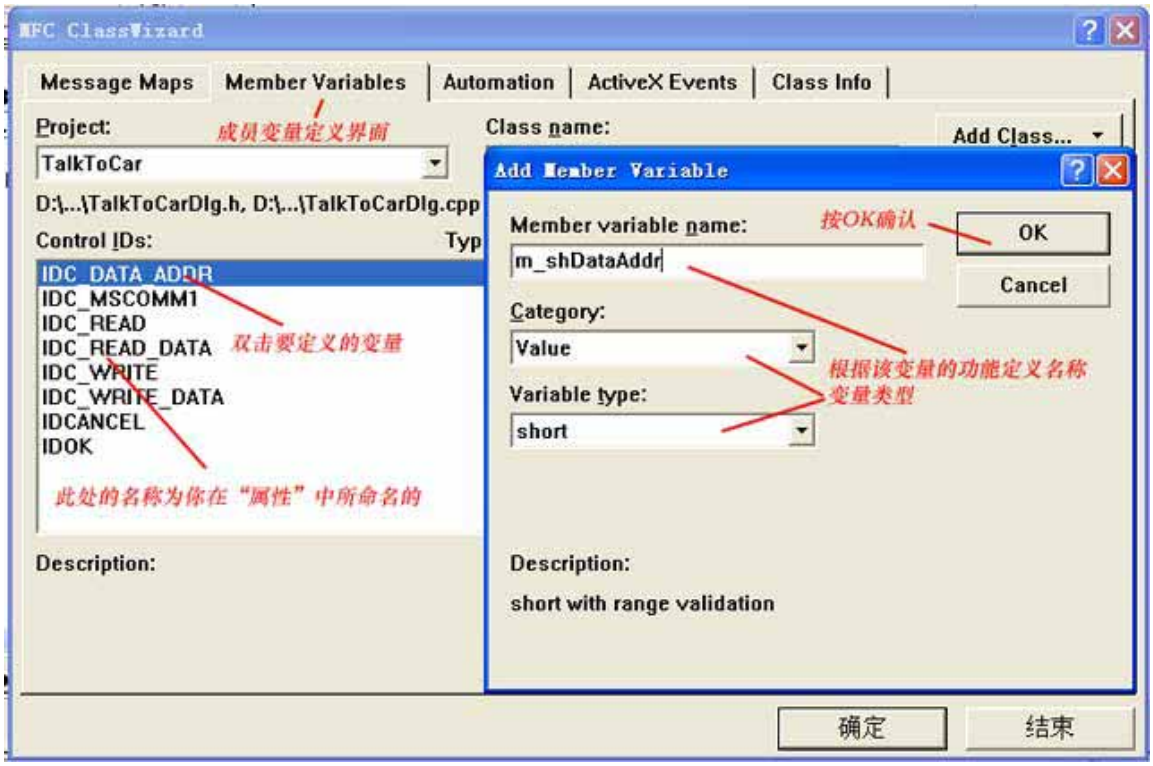
使用了如下四个素材：Static Text 、Combo Box、Edit Box、Button：

使用鼠标左键点住所需素材拖入对话框即可插入，可用鼠标右键激活“属性”，修改其名称及一些属性，如显示位置、只读、按钮名称等，注意 ID 的命名，建议保留原有的 IDC_，将后面改为可以表达对应功能的名称。

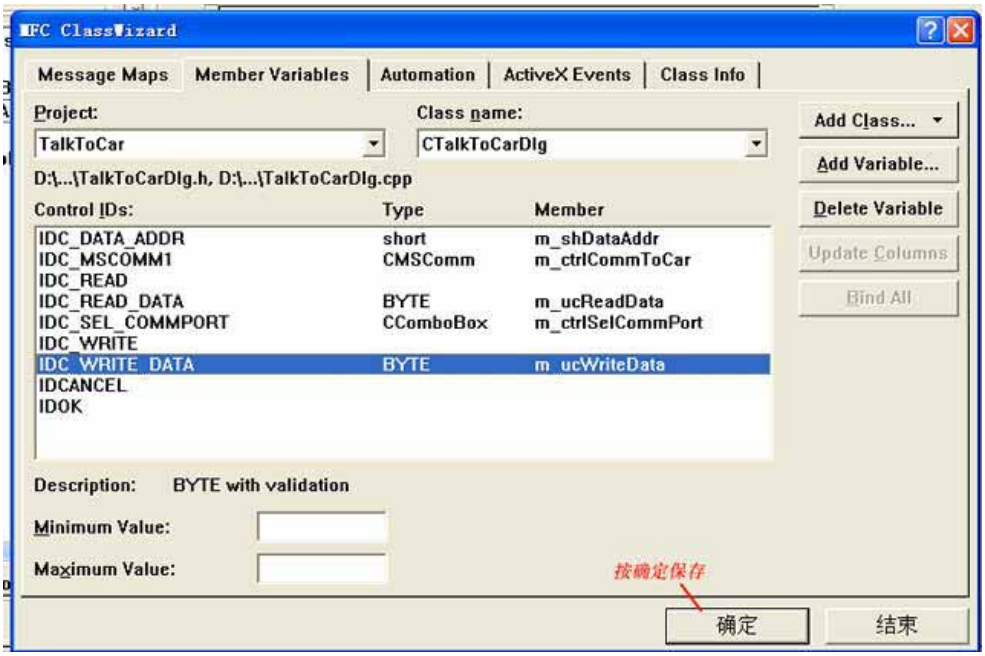
因为需要和小车通讯，所以必须将添加的通讯控件插入对话框中，插入方法一样，放在什么地方无所谓，它运行时不会显示出来：

将所有插入的素材利用“ 属性 ”功能修改为合适的名字，开始实质性编程 —— 定义成员变量，相当于以往编程的变量定义，因为在面向对象概念中，类等同于变量，包含了原来概念中的变量和函数，所以此处按钮、通讯处理均先作为变量处理，与原来的结构定义类似，只是结构的成员不只是数据，还有方法（即函数）。

将鼠标停留在对话框内，用右键进入“ 建立类向导 ”，显示：

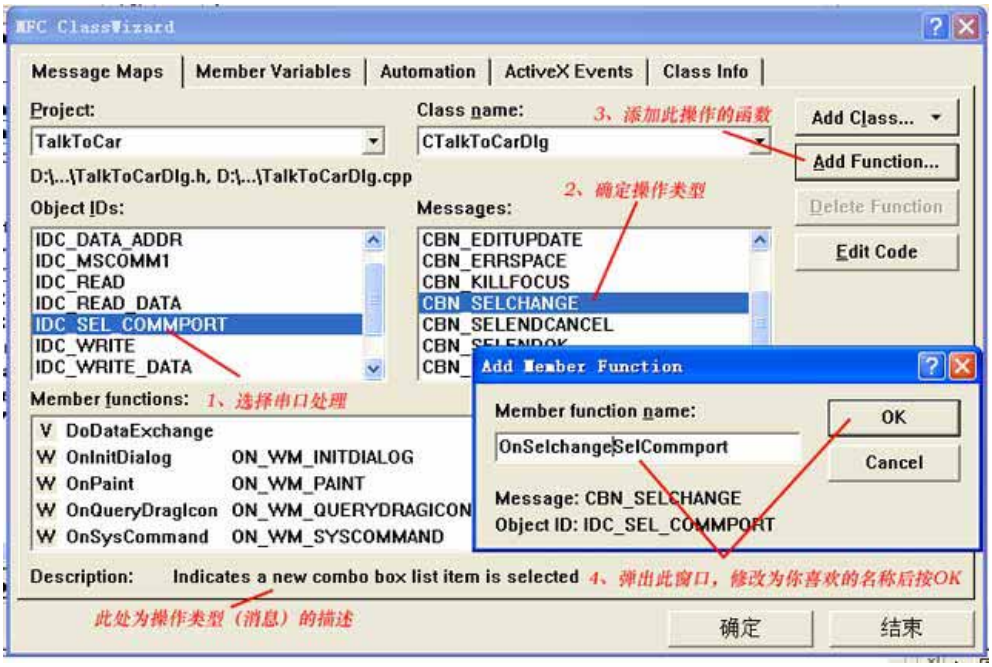


双击要定义的变量即跳出右侧小窗口，根据需要设定逐一设定，几个按钮不需要设定，但是通讯控件 IDC_MSCOMM1 需要定义。全部定义后如下：



此时，变量已定义完成，可以在“ ClassView ”中的“ CTalkToCarDlg ”类中看见这些定义好的变量，以及 VC 自动为这个类生成的基本函数，详细内容读者可自行尝试、发掘其奥妙，在此不再赘述。

下面开始根据需要添加一些为实现功能而用的函数，仍按上述方式进入“ 建立类向导 ”，进入后选择“ Message Maps ” 页面：



上图中的操作完成了“ 选择串口 ”功能的函数框架建立。Windows 程序是消息驱动的，

上图所注释的“ 确定操作类型 ”是俗称，准确的表述应该是“ 消息 ”，图中所选择的是“ Combo Box 中选择改变 ” 消息。

按 “ Edit Code ” 即可自动转到源程序编写界面，可以看到如下代码：

```
void CTalkToCarDlg::OnSelCommPort()
{
   // TODO: Add your control notification handler code here
    （在此处添加相应的处理程序）
}
```

按此方式可以将所有功能对应的函数框架构建好。主要有：读、写数据按钮，确定、取消按钮暂无用处，不去管它。

但是有一个函数比较特别，就是通讯接收处理，选择 “ ObjectID ” 窗口中的 IDC_MSCOMM1，在 “ Message ” 中只有一个选项 “ OnComm ”，添加这个函数，这实际上和单片机中的中断处理类似，它由串口操作的消息所激励，如收到一个字节、发送完等，详细地消息定义请看 VC 的帮助文件。

此时，基本程序框架已完成，可以在函数框架中加入自己的处理程序，和以往的 C 语言编程没有太大的差别，有些细节可以看程序中的注释。注意，所有初始化变量的功能添加在 “ CTalkToCarDlg::OnInitDialog() ” 中，在提示：

// TODO: Add extra initialization here 后插入。

更详细的串口通讯说明可以参阅参考资料 1 的第二章。

此程序的全部源代码见附件，作为入门，自我感觉应该选用最简单的例子，否则过多的内容会扰乱思路，反而无法理解了。读者可以在此基础上修改、完善，添加一些功能，如结合第一篇中的例子，通过 PC 机灵活改变用莫尔斯电码发送的字符串。

程序主要内容在 “ CTalkToCarDlg.CPP ” 及对应的头文件中。

三、 交流的用途

之所以花如此精力去讨论如何实现与小车交流，是为了让后面做事时能有一个好用的工具，“工欲善其事，必先利其器”，这两篇所描述的内容都是为了进一步控制小车所准备的。

第一篇实际上是为小车提供了一个直接的**人机界面**，不丰富但简单，而这一篇借用 PC 机提供了一个可以任意驾驭、丰富多彩的**人机界面**，而且是“双向”的，只是略显复杂。

搞单片机的都知道，需要用仿真机调试，但一方面“钱”是个障碍，同时仿真机的使用学习及资源占用也是问题，而且每搞一种单片机都要配置仿真机就更不经济，所以学会自己在程序中嵌入“Debug”功能，借助 PC 机实现基本的调试是十分有用的。

再者，很少有仿真能够通过无线实现的，不管是 JTAG 还是 BDM，用仿真头的就更不可能了。

读、写内存是最常用的调试功能之一，可以帮助你监测程序运行的状态和结果，修改相应的变量以改变程序的运行状态。

我看过多次大学生机器人赛，在轨迹赛中常看见选手一次次的修改参数去试，如果能够检测到运行遇到问题时相应的变量如何，那他们修改参数将会大大减少盲目性。

借助于交流功能，还可以让小车在运行中连续输出主要的参数，如轨迹赛时的光感检测值、电机控制值等，以便随时掌握小车状态，为优化对策提供极大的帮助。

四、 结语

本篇不仅为初学者提供了编写串口程序的示范，还期望帮助学单片机的读者也学会基本的 PC 机编程。

可以比较一下两侧的程序，可以发现其处理核心程序是多么相似？差别只在于程序的框架，在 PC 机上，VC 帮你完成了，而单片机需要自己构建。

但是，这一切要建立在使用 C 语言编程上，仔细阅读提供的单片机程序，你将发现它与使用什么单片机关系并不大，在处理部分更是如此，如果将变量名统一，单片机上的接收处理程序都可以直接拷贝到 PC 上。

这就是我想表达的：学习单片机不必过于关注单片机是什么，控制的逻辑和算法才是主要的。汇编可以帮助你理解单片机的工作原理，但尽量不要用其编程。

学会用适当的方式（如：宏定义）化解硬件的困扰，使得程序尽量不依附硬件，这样一方面可以提高程序的可读性、可靠性，另一方面使得你的程序可以移植到任何单片机上，使你的成果可以作为日后的基础，不至于成为海滩上的“ 砂雕 ”！

不知读者注意没有，两次程序中用了大量的符号化常量，并且变量名也很长，我看过一些初学者的程序，变量命名十分随意，而且程序中直接使用常数，这都不是好习惯。常量问题会带来日后程序修改的难度，而变量名不规范程序难读，不易发现问题。

附件一是我根据《代码大全》一书约定的一个命名规则，供大家参考。

记住自己做此事的目的 —— 学习，不要过于关注结果，要注重实现结果的过程和手段，培养良好的编程习惯。我强烈建议读者购买此书，经常查阅。

基本工具已准备好，之三开始涉入正题：让小车动起来！

附：

1、PC 机上的通讯示例程序 TalkToCar （压缩包，包含该工程目录下的全部内容）

2、单片机上的通讯示例程序，压缩包，包含两个程序：

YM_Prog - 2.C —— 在 YM_Prog - 1B.C 基础上添加通讯功能；

YM_Prog – 2A.C —— 在 YM_Prog - 2.C 上增加用主控指示灯 DEBUG 的示范。

附件一：

C51 程序变量命名规则：

常量: 大写字母

变量: 由两部分组成，用“ _ ”分开，前半部分说明作用域和功能，后半部分用小写字母说明类型，之后是一个大写字母开头的变量名称，如下：

（作用域）（功能）_ （数据类型）（大写字母开始的名称）

作用域：

1、全局变量: g

2、模块变量(在一个模块中公用的变量): m

局部变量无此信息。

功能：

1、数祖: a

2、数组下标: i

3、指针: p

4、计数器： c

5、枚举: e

数据类型：

1、字符： ch

2、无符号字符 uc

3、整型： i

4、无符号整型： ui

5、长整型： l

6、无符号长整型： ul

7、浮点数： f

8、无符号浮点数： uf

9、位变量： b

10、自定义类型变量: s

参考资料：

1、《Visual C++ 串口通信工程开发实例导航》 ISBN7-115-10949-4

2、《代码大全》第二版 ISBN 7 – 121 – 02298 – 2（第一版好像网上有）