# 第 8 章“组态，为王”
## 应用软件工具之一

## 8.1 组态软件简介

组态软件，又称组态监控系统软件。译自英文 SCADA,即 Supervisory Control and Data Acquisition（数据采集与监视控制）。它是指一些数据采集与过程控制的专用软件。它们处在自动控制系统监控层一级的软件平台和开发环境，使用灵活的组态方式，为用户提供快速构建工业自动控制系统监控功能的、通用层次的软件工具。组态软件的应用领域很广，可以应用于电力系统、给水系统、石油、化工等领域的数据采集与监视控制以及过程控制等诸多领域。

组态软件在国内是一个约定俗成的概念，并没有明确的定义，它可以理解为“组态式监控软件”。“组态（Configure）”的含义是“配置”、“设定”、“设置”等意思，是指用户通过类似“搭积木”的简单方式来完成自己所需要的软件功能，而不需要编写

计算机程序，也就是所谓的“组态”。它有时候也称为“二次开发”，组态软件就称为“二次开发平台”。“监控（Supervisory Control）”，即“监视和控制”，是指通过计算机信号对自动化设备或过程进行监视、控制和管理。

组态软件是有专业性的。一种组态软件只能适合某种领域的应用。组态的概念最早出现在工业计算机控制中，如：DCS(采集控制系统）组态、PLC（可编程控制器）梯形图组态；人机界面生成软件就叫工控组态软件。在其他行业也有组态的概念，如AutoCAD，PhotoShop 等。不同之处在于，工业控制中形成的组态结果是用在实时监控的。从表面上看，组态工具的运行程序就是执行自己特定的任务。工控组态软件也提供了编程手段，一般都是内置编译系统，提供类 BASIC 语言，有的支持 VB，现在有的组态软件甚至支持 C#高级语言。

组态软件大都支持各种主流工控设备和标准通信协议，并且通常应提供分布式数据管理和网络功能。对应于原有的 HMI（人机接口软件，Human Machine Interface）的概念，组态软件还是一个使用户能快速建立自己的 HMI 的软件工具或开发环境。在组态软件出现之前，工控领域的用户通过手工或委托第三方编写

HMI 应用，开发时间长，效率低，可靠性差；或者购买专用的工控系统，通常是封闭的系统，选择余地小，往往不能满足需求，很难与外界进行数据交互，升级和增加功能都受到严重的限制。组态软件的出现使用户可以利用组态软件的功能，构建一套最适合自己的应用系统。随着它的快速发展，实时数据库、实时控制、SCADA、通讯及联网、开放数据接口、对 I/O 设备的广泛支持已经成为它的主要内容，监控组态软件将会不断被赋予新的内容。

“组态”的概念是伴随着集散型控制系统（Distributed Control System 简称 DCS）的出现才开始被广大的生产过程自动化技术人员所熟知的。在工业控制技术不断发展和应用的过程中，PC（包括工控机）相比以前的专用系统具有的优势日趋明显。这些优势主要体现在：PC 技术保持了较快的发展速度，各种相关技术已经成熟；由 PC 构建的工业控制系统具有相对较低的拥有成本；PC 的软件资源和硬件资源丰富，软件之间的互操作性强；基于 PC 的控制系统易于学习和使用可以容易地得到技术方面的支持。在 PC 技术向工业控制领域的渗透中，组态软件占据着非常特殊而且重要的地位。

常见的国外组态软件主要有：

1.InTouch

2.IFix

3.Citech

4.WinCC

5.ASPEN-tech

6.Movicon

7.GENESIS 64

其中最早进入国内的是 InTouch，也是最具代表性的组态软件之一。

国内常用的组态软件有：

1.世纪星

2.三维力控

3.组态王 KingView

4.紫金桥 Realinfo

5.MCGS

6.态神

7.uScada

8.Controx（开物）

9.E-Form++组态源码解决方案

10.iCentroView

11.QTouch

其中组态王 KingView 的市场占有率最高。

随着工业自动化水平的迅速提高，计算机在工业领域的广泛应用，人们对工业自动化的要求越来越高，种类繁多的控制设备和过程监控装置在工业领域的应用，使得传统的工业控制软件已无法满足用户的各种需求。在开发传统的工业控制软件时，当工业被控对象一旦有变动，就必须修改其控制系统的源程序，导致其开发周期长；已开发成功的工控软件又由于每个控制项目的不同而使其重复使用率很低，导致它的价格非常昂贵；在修改工控软件的源程序时，倘若原来的编程人员因工作变动而离去时，则必须同其他人员或新手进行源程序的修改，因而更是相当困难。通用工业自动化组态软件的出现为解决上述实际工程问题提供了一种崭新的方法，因为它能够很好地解决传统工业控制软件存在的种种问题，使用户能根据自己的控制对象和控制目的的任意组态，完成最终的自动化控制工程。

组态（Configuration）为模块化任意组合。通用组态软件主要特点：

（1）延续性和可扩充性。用通用组态软件开发的应用程序，当现场（包括硬件设备或系统结构）或用户需求发生改变时，不需作很多修改而方便地完成软件的更新和升级；

（2）封装性（易学易用），通用组态软件所能完成的功能都用一种方便用户使用的方法包装起来，对于用户，不需掌握太多的编程语言技术（甚至不需要编程技术），就能很好地完成一个复杂工程所要求的所有功能；

（3）通用性，每个用户根据工程实际情况，利用通用组态软件提供的底层设备（PLC、智能仪表、智能模块、板卡、变频器等）的 I/O Driver、开放式的数据库和画面制作工具，就能完成一个具有动画效果、实时数据处理、历史数据和曲线并存、具有多媒体功能和网络功能的工程，不受行业限制。

组态软件指一些数据采集与过程控制的专用软件，它们是在自动控制系统监控层一级的软件平台和开发环境，能以灵活多样的组态方式（而不是编程方式）提供良好的 用户开发界面和简捷的使用方法，它解决了控制系统通用性问题。其预设置的各种软件模块可以非常容易地实现和完成监控层的各项功能，并能同时支持各种硬件厂 家的计算机和 I/O 产品，与高可靠的工控计算机和网络系统结合，可向控制层和管理层提供软硬件的全部接口，进行系统集成。

组态软件通常有以下几方面的功能：

（1）强大的界面显示组态功能。目前，工控组态软

件大都运行于 Windows 环境下，充分利用 Windows 的图形功能完善界面美观的特点，可视化的 m 风格界面、丰富的工具栏，操作人员可以直接进人开发状态，节省时间。丰富的图形控仵和工况图库，既提供所需的组件，又 是界面制作向导。提供给用户丰富的作图工具，可随心所欲地绘制出各种工业界面，并可任意编辑，从而将开发人员从繁重的界面设计中解放出来，丰富的动画连接 方式，如隐含、闪烁、移动等等，使界面生动、直观。

（2）良好的开放性。社会化的大生产，使得系统构成的全部软硬仵不可能出自一家公司的产品，“异构”是当今控制系统的主要特点之一。开放性是指组态软件能与多种通信协议互联，支持多种硬仵设备。开放性是衡量一个组态软件好坏的重要指标。组态软件向下应能与低层的数据采集设备通信，向上能与管理层通信，实现上位机与下位机的双向通信。

（3） 丰富的功能模块。提供丰富的控澜功能库，满足用户的测控要求和现场荽求。利用各种功能模块，完成实时监控 产生功能报表 业示历史曲线、实时曲线、提侠报警等功能，使系统具有良好的人机界面，易于操作，系统既叫适用于单机集中式控制、DCS 分布式控制，也可以是带远程遇信能 力的远程测控系统.

（4）强大的数据库。配有实时数据库，可存储各种数据，如模拟量、离散童、字符型等，实现与外部设备的数据交换。

（5）可编程的命令语言。有可编程的命令语言，使用户可根据自己的需要编鸾程序，蹭强图形界面

（6）周密的系统安全防范，对不同的操作者，赋予不同的操作权眼，保证整个系统的安全可靠运行。

（7）仿真功能．捉供强大的仿真功能使系统并行设计，从而缩短开发周期。

## 8.2 组态软件与 CAN 总线——强强结合

一方面组态软件得到了广泛的应用，另一方面 CAN 现场总线成本降低而逐步普及，使其有能力替代传统的 485＋modbus 应用模式，因此将 CAN 现场总线融入组态软件中是发展的大趋势，其对应用系统的性能提升起着非常积极的作用，是强强结合的产物。

然而，现实却与我们美好的愿望存在着很大差距，由于 CAN 应用层协议中的两个巨无

霸 CANopen 和 deviceNet 解析起来非常困难，使组态软件中相应的“驱动”也更加难于实现目前国内主流的组态软件对 CANopen 和 deviceNet 都没有有效的支持，国外组态软件有限的部分支持 CANopen 和 deviceNet 协议，且价格不菲。这给组态软件与 CAN 现场总线的结合带来了巨大的屏障。

TTCANopen 应用层协议的出现使我们看到了曙光。我们知道所有的组态软件对串口和 Modbus 的支持是与生俱来的，由于 TTCANopen 应用层协议的直读性和其类 Modbus 特性以及“转换模块”的“虚拟串口”特性，使 TTCANopen 的解析和组态软件驱动的开发的难度与 Modbus 站在了同一“起跑线”上，由此铺平了组态软件与 CAN 现场总线结合的道路，大凡能够开发串口设备、Modbus 驱动的程序员便可非常容易的开发出 TTCANopen 驱动。

为了给用户提供一个样例，起着抛砖引玉

的作用，我们聘请了北京中铁航机电公司的工程师，尝试开发了一个“组态王”组态软件驱动。

## 8.3 组态软件驱动程序的开发

组态软件驱动主要分为“设备驱动”和“协议驱动”。“设备驱动”是针对某些具体设备开发的驱动程序，它只对这些设备有效；“协议驱动”是针对某种通用协议编写的驱动凡是符合协议规范的设备都会被组态软件所支持，如“Modbus”驱动，我们要开发的样例驱动属于后者，是针对 TTCANopen 初级篇中的协议规范的。

组态软件驱动与我们通常所说的系统设备驱动是不同的。一般“通讯设备”首先是通过系统设备驱动完成其硬件与操作系统之间的交互关系，再上面一层才是组态软件驱动，完成通讯数据的解析使其能够被组态软件所识别和使用。确切的说组态软件驱动是一个“协议解

析器”，它将底层通讯设备按某种协议传输的数据解析为组态软件的特征变量，反之亦然。

在开发组态软件驱动前，我们给国内几个主要组态软件开发商打通了电话，索取其组态软件驱动开发包，其中，只有北京亚控公司发来了“组态王”的驱动开发包，还包括了一个长达 2 小时的视频开发教程及相关文档，这对我们的帮助很大。这是我们第一次组织开发组态软件驱动程序，必然存在很多的经验不足和偏差，在这里我们将开发的主要过程展现给用户，也是一个相互学习的过程，当然我们更希望将来的驱动不是由我们来开发，而是由各组态软件公司开发，因为只有生产商才最了解自己产品的内核，开发出来的驱动才会更加稳定

## 8.3.1 组态王驱动开发环境

| | |
|---|---|
| 操作系统 | windows xp（sp3） |
| 编程语言 | visual studio.net 2003 |
| 开发软件包 | 组态王驱动开发包 3.0.0.7 |

面向协议　　TTCANopen 初级协议

## 8.3.2 安装组态王开发软件包

在安装组态王驱动开发软件包之前，我们应当安装好 visual studio.net 2003，然后运行“组态王驱动开发包 3.0.0.7（中文）.exe”见图 8－1，然后点击“下一步”选择安装目标文件夹见图 8－2，安装完成。组态王驱动开发工具能够自动生成驱动代码框架。

图 8－1 启动安装组态王驱动开发包


图 8－2 选择安装目标文件夹

## 8.3.3 生成基本的驱动程序代码

安装完组态王驱动开发包后，就可以启动visual studio.net 2003，生成基本的驱动程序框架。其具体步骤如下：

（1）启动 visual studio.net 2003；

（2）新建一个工程

“项目类型”选择“Visual C++项目”，

“模板”选择 “KingView Wizard”，

在下面的编辑框中输入工程的名字以及存储路径，

点击“确定”；

（3）生成一个驱动创建向导，该向导分为三页，第一页见图 8－3，是一个简要说明，没有任何选项和输入内容；

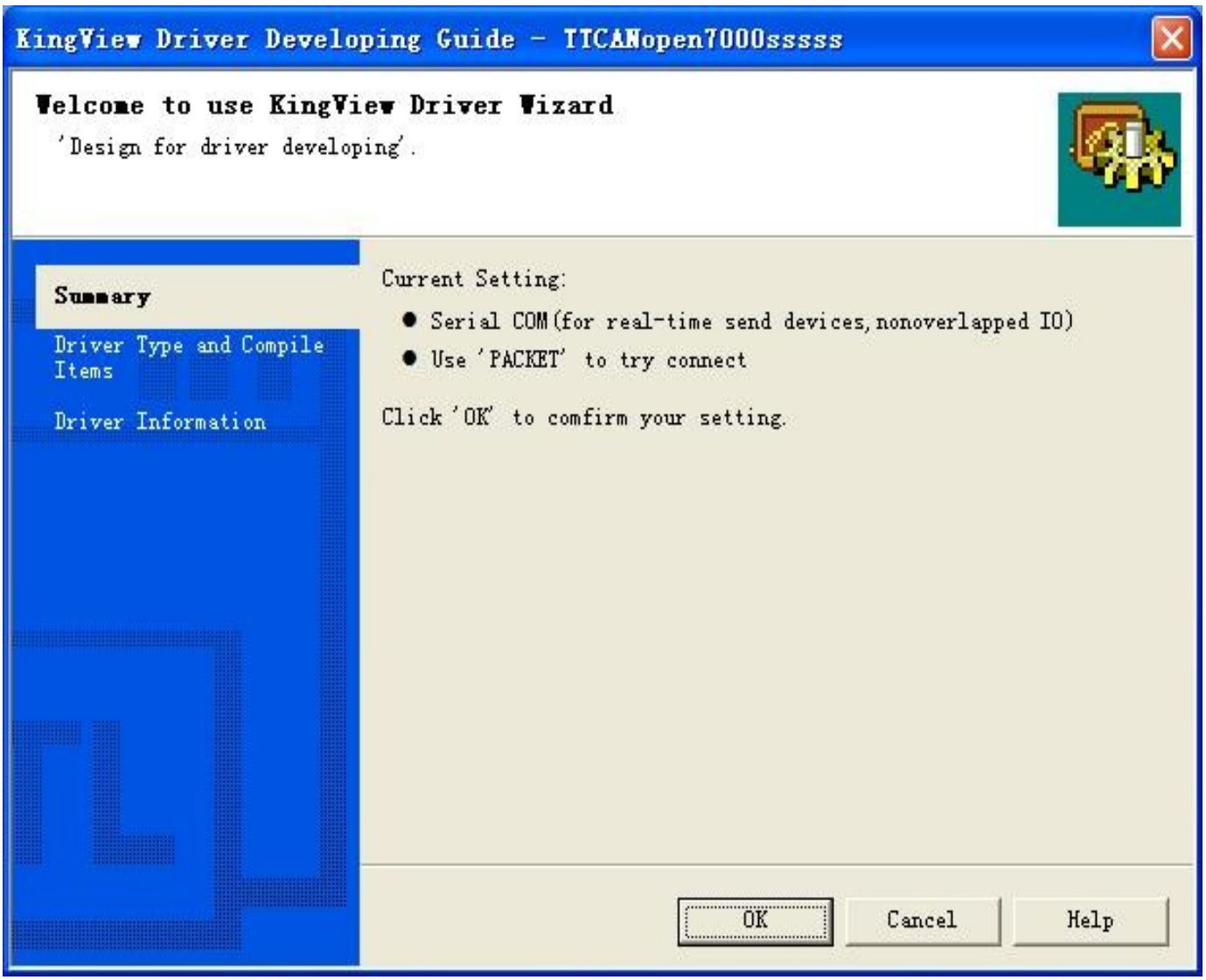


图 8－3 驱动简要说明

第二页见图 8－4，是驱动类型和编译选项，其中驱动类型有两个串口可选项和两个网络可选项，在这里我们选择 Serial COM(for real-time send devices, nonoverlapped IO )这种方式，其适用于下位设备实时上发数据的情况，驱动中

需要创建一个线程来监视串口事件的通讯类型采用非重叠 IO 方式，该方式能够支持 TTCANopen 中的事件触发和主动上报信息模式，而不仅仅是轮询这种简单方式。

设备名称：默认的设备名称是“Name1”，可以自由填写，但须注意应该与设备列表中的设备名称保持一致，这里我们使用“TTCANopen7000ssssss”这个名字。

尝试连接：默认选用 use“PACKET”。

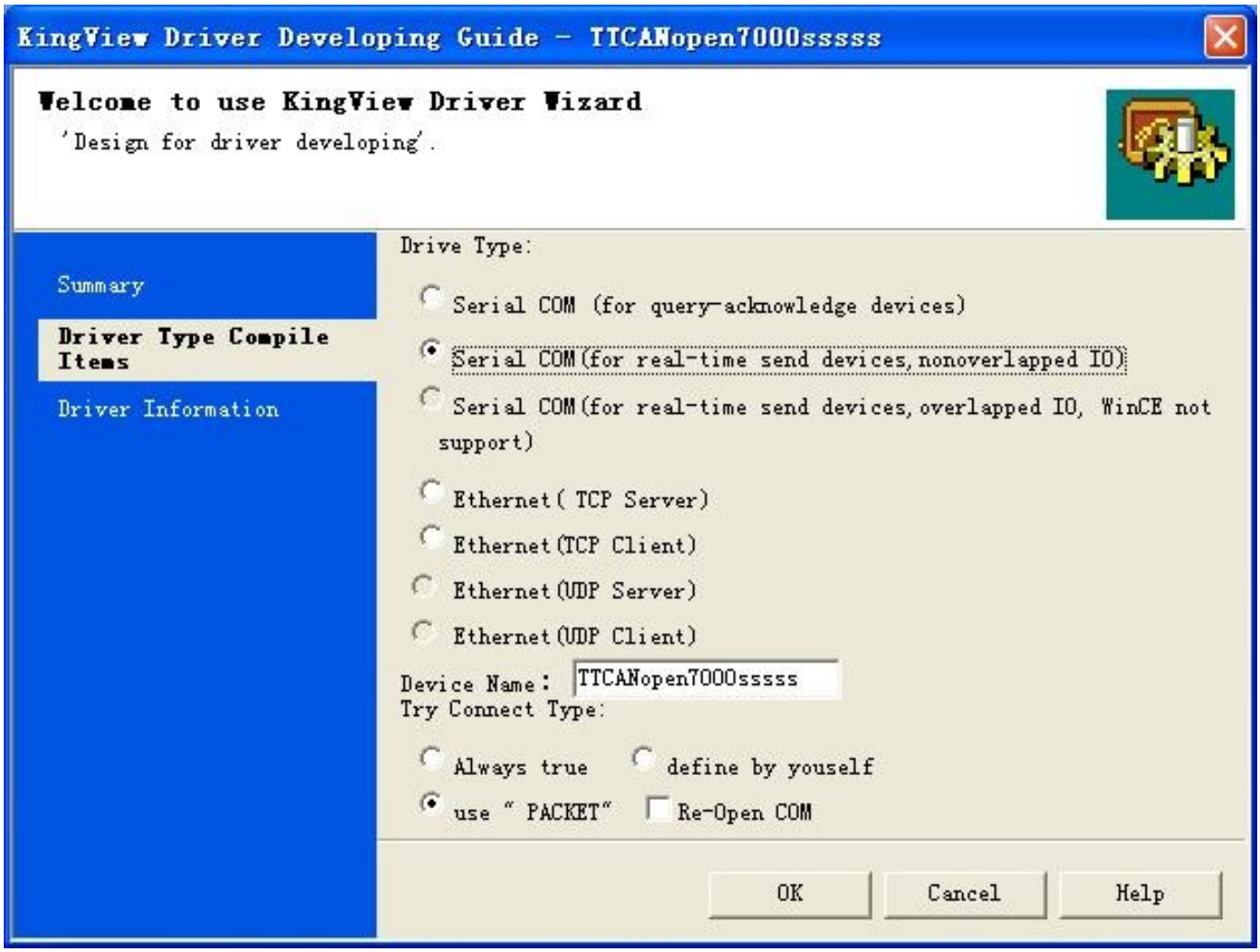

图 8－4 驱动类型和编译选项

第三页见图 8－5，是驱动基本信息，主要包括：驱动名称、版本、描述、程序员；其他支持：USB 通讯和 ADO 数据库操作；

点击“OK”即生成了基本的驱动框架，在这个框架中包含了许多的文件见图 8－6 中所示，其中“DevTTCANopen7000ssssss.cpp”这个文件非常重要，我们编写的驱动程序需要“填充”和“修改”的 80％的内容都在这个文件中完成。

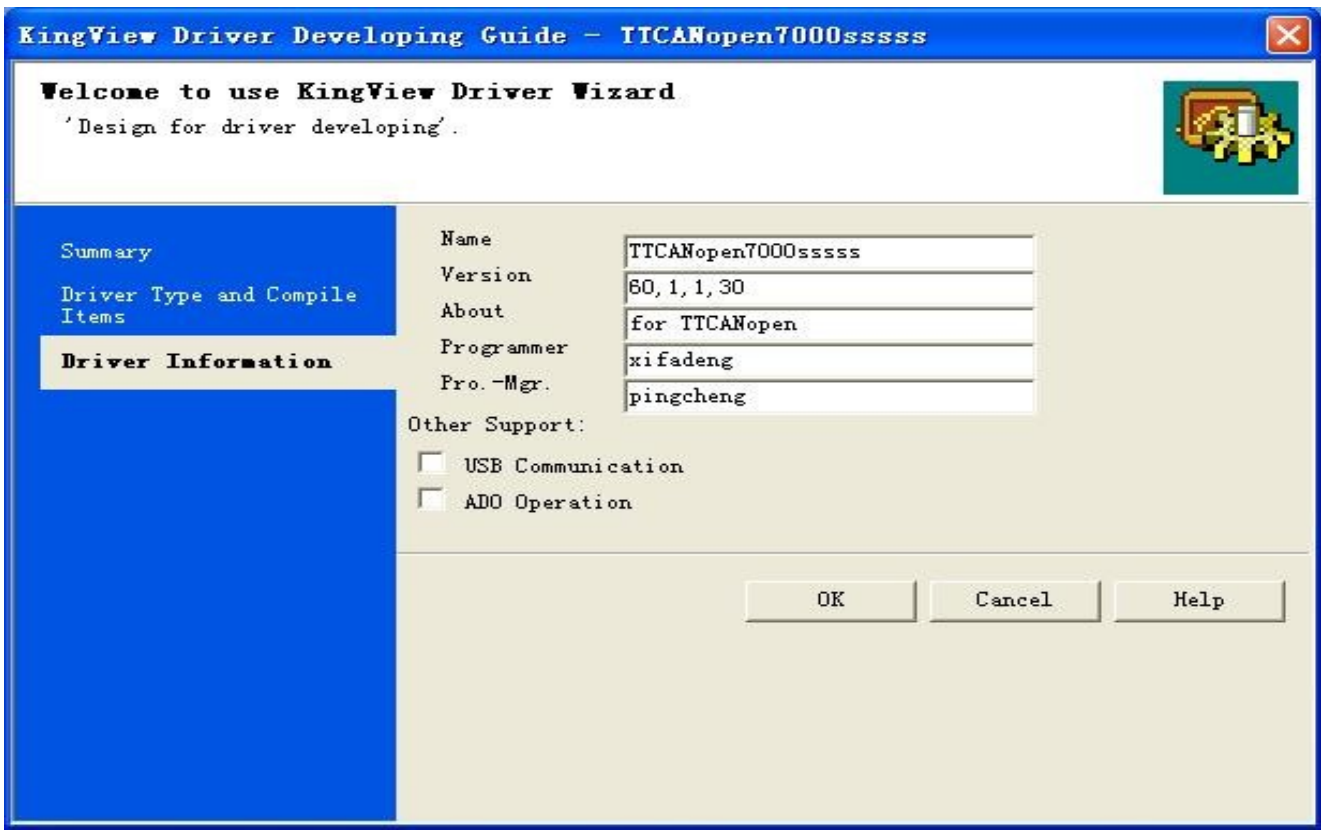

图 8－5 驱动基本信息

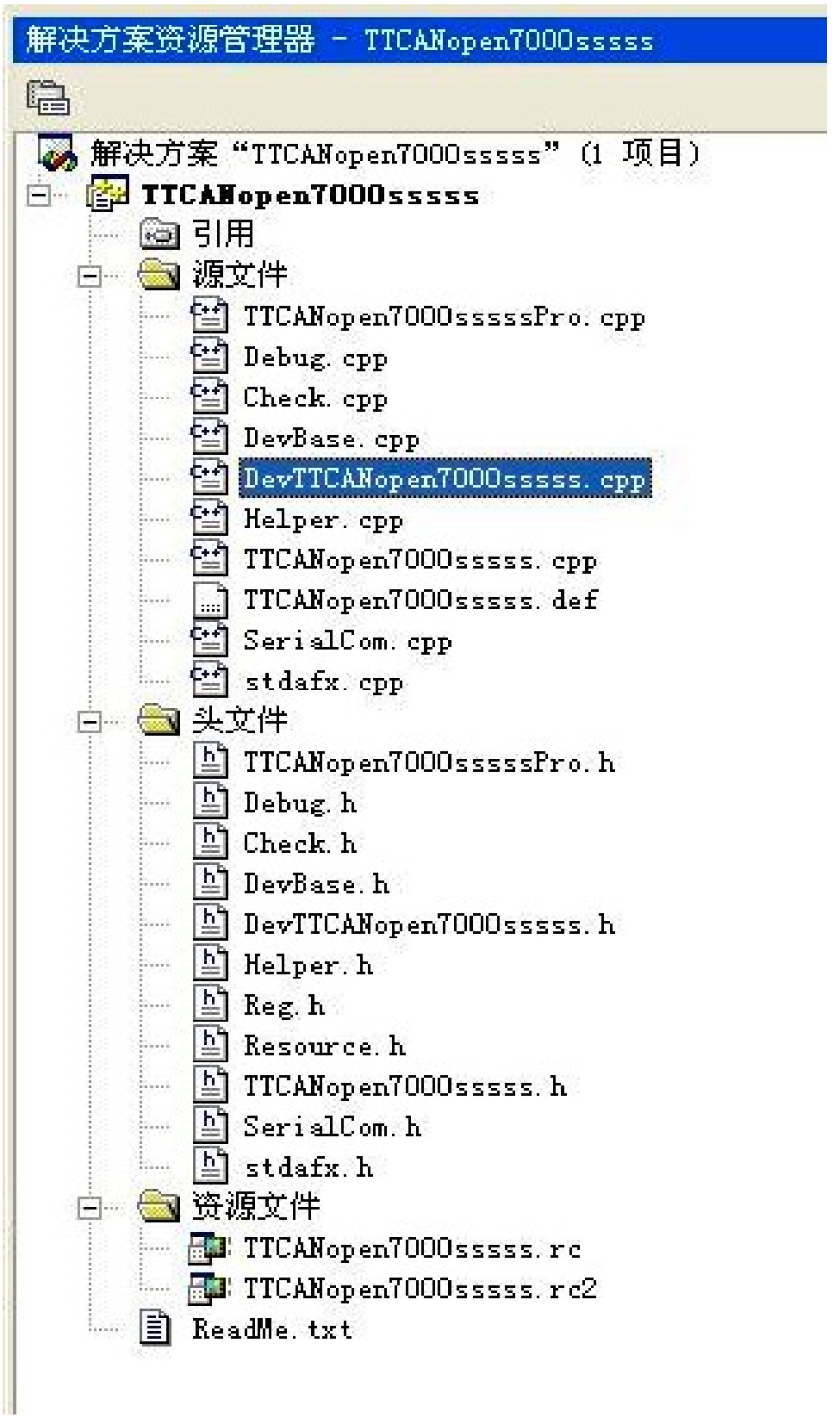


图 8－6 驱动框架中的文件

## 8.3.4 DevTTCANopen7000sssss.cpp 文件的“填充”和“修改”

### （1）组态软件中注册变量

在组态软件中根据应用的不同会用到各种变量，如：“位”变量、“字节”变量、“整数”变量、“长整数”变量和“浮点”变量等在 DevTTCANopen7000sssss.cpp 文件中专门有一个数据结构用于定义用户用到的变量类型、寄存器范围和变量属性（读/写）见图 8－7。

图 8－7　自动生成的用户变量结构列表

```
static REG_INFO gsRegInfo[]=
{
//add your code  here, the following for your reference
    {_T("FLOW"), 0, 0, FLOAT_DATATYPE, PT_READ },       // the current flow register, read only
    {_T("PARA"), 0, 2, FLOAT_DATATYPE, PT_READ }        //  parameter register, read only

};
```

（后续部分 略.....）

注：本章后续被邓先生重写，

见“新第 8 章　**组态 为王**”