通信接口及协议配置组件

软件用户手册

凯云联创(北京)科技有限公司

2018年10月

**目录**

[1 产品概述 1](#_Toc527640257)

[1.1 功能概述 1](#_Toc527640258)

[2 运行环境 1](#_Toc527640259)

[2.1.1 硬件环境 1](#_Toc527640260)

[2.1.2 软件环境 1](#_Toc527640261)

[2.2 软件安装 1](#_Toc527640262)

[3 软件使用介绍 1](#_Toc527640263)

[3.1 概述 1](#_Toc527640264)

[3.2 运行界面 2](#_Toc527640265)

[3.2.1 功能按钮 2](#_Toc527640266)

[3.2.2 代码编辑器 4](#_Toc527640267)

[3.3 生成代码集成方法 15](#_Toc527640268)

[3.4 软件配置使用示例 15](#_Toc527640269)

[4 相关文件 18](#_Toc527640270)

# 产品概述

通用接口及协议配置组件具有实现通信模块、配置文件编辑、解析、及自动生成程序代码功能。通信配置组件的输入内容是配置文件，输出内容是生成代码。

## 功能概述

1. 标识号:KY-TXJKJXYPZZJ
2. 标题:通信接口及协议配置组件用户手册
3. 说明:本文档适用于通信接口及协议配置组件软件的用户使用说明。

# 运行环境

### 硬件环境

由于本软件通信接口特性的特点，在应用时如果用到具体的板卡通信，那么软件过行时应要使相应的接口板卡处理完好就绪状态，以实现软件应用时的通信要求。

### 软件环境

1.WIN7操作系统

## 软件安装

# 软件使用介绍

## 概述

在具备软件运行环境的基础上，双击运行FrameIO.exe文件启动软件。

## 运行界面

### 功能按钮

功能按钮区主要包括“新建项目”、“打开项目”、“保存”、“切换编辑视图”、“显示/隐藏输出面板”、“剪切”、“复制”、“粘贴”、“撤销”、“重做”、“查找/替换”、“代码检查”、“输出”、“关于”

（1）新建项目

点击“新建项目”按钮，系统会弹出新建对话框，选择新建项目所在目录，输入文件名称，点击“保存”按钮，系统生成一个.flo扩展名的项目文件。

（2）打开项目

点击“打开项目”按钮，系统弹出打开文件对话框，选择一个已经存在的项目文件(.flo文件)，点击对话框上的“打开”按钮，系统打开一个已有的项文件。如果系统已经打开一个项目文件，在点击“打开项目”前会有是否保存原项目的提示。

（3）保存

点击“保存”按钮，将编辑的代码保存到新建或打开的项目文件中。

（4）切换

点击“切换编辑视图”将切换成界面编辑模式或脚本编辑模式。

（5）显示/隐藏输出面板

点击“显示/隐藏输出面板”界面上的输出信息窗口将被切换显示或隐藏。

（7）剪切

点击“剪切”按钮，将代码剪切到剪切板上。

（8）复制

点击“复制”按钮，将代码复制到剪切板上。

（9）粘贴

点击“粘贴”将剪切板上已复制的内容插入到当前光标的位置。

（10）撤销

点击“撤销”按钮，撤销当前最近一次编辑操作。

（11）重做

点击重做”按钮，重做当前最近一次编辑操作。

（12）查找/替换

点击“查找/替换”按钮，在代码编辑窗口中查找和替换指定的内容。

（13）代码检查

在代码编辑窗口中，编辑代码后，点击“代码检查”铵钮，执行代码检查，检查结果会在输出信息窗口中显示。

（14）输出

点击“输出”按钮，将信息输出到输出窗口中。

（15）关于

点击“关于”按钮，显示软件的名称及版本信息。

### 代码编辑器

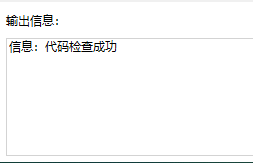
代码编辑器是系统的主要工具，用于编辑配置文件。相应的配置主要包括项目配置、分系统配置、数据帧配置、枚举项配置。

（1）标识符命名规则

标识符有统一的命名规则，都是以字线、数字、下划线的组合，其中首字符不能是数据的规则。如果不符合系统的命名规则要求，在标识符下面会以波浪线显示非法的命名规则。

（2）代码检查

编辑过程中和编辑结束可以点击“代码检查”，在下面的输出信息窗口会显示相应的代码检查结果的输出信息。如下图所示：



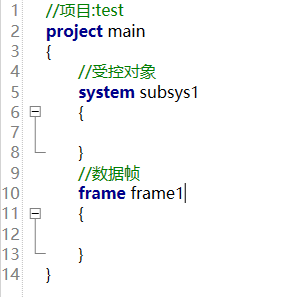
（3）关键字

系统中预定了一部分关键字，包括：

|  |  |  |
| --- | --- | --- |
| 编号 | 关键字标识 | 关键字描述 |
| 1 | project | 项目关键字 |
| 2 | system | 分系统关键字 |
| 3 | channel | 通道关键字 |
| 4 | action | 动作关键字 |
| 5 | frame | 数据帧关键字 |
| 6 | enum | 枚举类型关键字 |
| 7 | // | 注释符 |

#### 项目配置

进入系统后，点击“新建项目”按钮系统会弹出一个新建文件的对话框，输入文件名后，系统自动创建一个项目，代码编辑器中会自动显示初始的项目配置代码框架。如下图所示：



**图1配置代码框架**

代码框架中主要包括注释、项目名称、分系统名称和数据帧名称。说明如下：

1. 注释：注释以“//”开头。其后为注释内容，支持中文。

**语法：**//项目名称

1. 项目定义：项目名称以“**project**”关键字开头，其后为项目名称，项目名称支持修改，“**project**”关健字与项目名称之间以逗号分隔。项目定义中包括分系统定义、数据帧定义及枚举定义。

**语法：project** name

{

//分系统定义

//数据帧定义

//枚举定义

}

1. 分系统定义：分系统名称以“**system**”开头，其后为分系统名称，分系统名称命名规则遵循标识符命名规则。窗口下方的输出窗口中会有相应的提示信息。分系统定义包括属性定义、通道定义及操作定义。

**语法**：**system** name

{

//属性定义;

//通道定义;

//操作定义;

}

1. 数据帧定义：数据帧名称以“**frame**”开头，数据帧名称的命名规则与分系统的命名规则相同，并且可以与分系统重名。数据帧定义中包括数据字段定义。

**语法：frame** name

{

//字段定义;

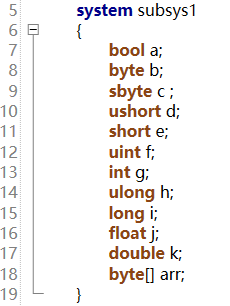
}

#### 分系统配置

分系统配置主要包括通道配置、属性配置和操作配置。

1. 属性

分系统属性包括12种数据类型，需要指定属性名称和数值类型，分别为bool、byte、sbyte、ushort、short、uint、int、ulong、long、float、double及以上类型的数组。属性的名称为一个字符串标识，以字母开头，后跟若干个下划线、字母或数字的组合，通信配置组件使用者需确保该名称标识与业务系统中的标识没有冲突。编辑器中书写如下图所示：



**图2 分系统属性示意图**

数据类型说明如下表:

|  |  |  |
| --- | --- | --- |
| 编号 | 标识 | 数据类型说明 |
| 1 | bool | 布尔 |
| 2 | sbyte | 8位带符号整型 |
| 3 | byte | 字节 |
| 4 | ushort | 16位无符号整型 |
| 5 | short | 16位带符号整型 |
| 6 | uint | 32位无符号整型 |
| 7 | int | 32位带符号整型 |
| 8 | ulong | 64位无符号整型 |
| 9 | long | 64位带符号整型 |
| 10 | float | 32位单精度浮点型 |
| 11 | double | 64位双精度浮点型 |
| 12 | [] | 数组类型 |

1. 通道

通道是受控对象传输数据时使用的硬件资源，向受控对象配置项中增加通道配置时，需要指定通道的类型、通道名称及通道参数。

可选的通道类型有：can、com、tcpserver、tcpclient、udp、di、do，通道名称是使用者指定的字符串标识，每一种通道类型都有各自不同的通道参数。

* 通道关键字：

以关键字“channel”开关，后面为通道的名称，中间以空格分隔。

* 通道名称标识符：

其中通道名称和通道类型之间以冒号分隔，通道类型高亮显示。

**语法：channel** ch1 :**can**

{

参数名称：参数值

}

各通道参数配置如下：

* 研华can类型通道的参数

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 标识 | 取值 |
| 1. | 制造商 | vendor | yh（研华） |
| 2. | 通道号 | channelind | 正整数 |
| 3. | 波特率 | baudrate | 正整数 单位为**K**bit/s |
| 4. | 发送超时 | waittimeout | 无符号整数 |

* 广州致远can类型通道的参数

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 标识 | 取值 |
| 1. | 制造商 | vendor | gzzy（广州致远） |
| 2 | 设备类型 | devtype | 正整数 |
| 3. | 设备号 | devind | 设备号 |
| 4. | 通道号 | channelind | 正整数 |
| 5. | 波特率 | baudrate | 正整数 单位为**K**bit/s |
| 6. | 验收码 | acccode | 无符号整数 |
| 7. | 屏蔽码 | accmark | 无符号整数 |
| 8. | 工作模式(正常工作/自发自收) | mode | 字节 |
| 9. | 过滤模式(单滤波/双滤波) | filter | 字节 |
| 8. | 发送超时 | waittimeout | 无符号整数 |

* com口类型通道的参数

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 标识 | 取值 |
| 1. | 端口 | portname | 字符型 |
| 2. | 波特率 | baudrate | 无符号整数 单位为**K**bit/s |
| 3. | 奇偶校验 | parity | 无符号整数 |
| 4. | 数据位长 | databits | 无符号整数 |
| 5. | 停止位 | stopbits | 无符号整数 |

* tcpserver通道的参数

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 标识 | 取值 |
| 1. | 监听IP | serverip | 字符串 |
| 2. | 监听端口 | port | 无符号整数 |
| 3 | 待测设备地址 | clientip | 字符串 |

* tcpclient通道的参数

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 标识 | 取值 |
| 1. | 远程IP地址 | serverip | 字符串 |
| 2. | 端口 | port | 无符号整数 |

* upd通道的参数

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 标识 | 取值 |
| 1. | 本地地址 | localip | 字符串 |
| 2. | 本地端口 | localport | 无符号整数 |
| 3 | 远端地址 | remoteip | 字符串 |
| 4 | 远端端口 | remoteport | 无符号整数 |

* di通道的参数

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 标识 | 取值 |
| 1. | 设备号 | deviceno | 无符号整数 |
| 2. | 通道号 | channelno | 无符号整数 |
| 3 | 最小信号值 | minvalue | 无符号整数 |

* do通道的参数

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 名称 | 标识 | 取值 |
| 1. | 设备号 | deviceno | 无符号整数 |
| 2. | 通道号 | channelno | 无符号整数 |
| 3. | 最小信号值 | minvalue | 无符号整数 |

1. 操作(action)

分系统的操作包括三种，分别为发送(send)、接收(recv)。

操作以关键字“action”开头，后面跟着是操作的名称标识，名称标识后以冒号定义操作类型，操作作名后面以空格分隔定义操作所用的数据帧名称，数据帧名称后面以空格分隔定义操作所在的通道。注意:action执行体里定义可以定义@开头的语句赋值代码,解析程序不对@开头的语名进行解析。

**语法：**  **action** ac2:**send** frame2 **on** aaa

        {  
            seg1:c;  
            seg2:d;  
        }

**@用法：**  假如定义了一个数据帧如下：

**Frame**SendFrame\_Status

**{**

**integer** g\_header\_datatype**bitcount=**8;

**integer** g\_header\_PowerSupplyIdx**bitcount=**8;

**integer** g\_Status1**bitcount=**1;

**integer** g\_retained**bitcount=**7;

**integer** g\_Status1**bitcount=32 signed=ture**;

**}**

写发送动作时，action的写法如下，@后面的语句将**g\_header\_datatype**赋值为2**。**

**action** ac2:**send**SendFrame\_Status**on** aaa

        {

**@data.g\_header\_datatype=2;**            seg1:c;  
  }

#### 数据帧配置

数据帧配置项用以配置传输协议中规定的报文数据格式，数据帧配置项是分系统完成数据输入输出的必备条件；每一个数据帧配置项均会产生与之对应的数据帧类，该类内部封装了报文解析的功能。

增加数据帧配置项时只需要指定数据帧的名称即可，数据帧配置项的具体内容由一组字段配置项构成，字段配置项按照字节流顺序依次配置。每项字段配置项需要指定字段名称、字段类型、字段属性及字段相关操作。数据帧的定义格式在项目配置相关章节已做介绍，这里主要详细说明数据帧字段。

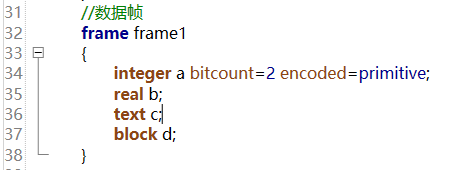


图4 数据帧编辑示意图

（1）字段类型与属性

数据帧中的字段类型有：integer、real、block三种；分别代表整型、浮点数、字符串和字节块，每种类型均有各自不同的属性。虽然只有四种字段类型，但通过与字段属性的灵活组合，可以配置出各种复杂的报文格式。

每种字段类型均具有多种字段属性，用以配置报文解析的各种不同方式，所有属性均为可选配置项，属性的默认值通常都是协议解析中最常用的选择方式。

* Integer字段

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 标识 | 是否属性 | 描述 |
| 1. | signed | Y | 属性标识整数是否有符号,默认 值为false |
| 2. | bitcount | Y | 属性标识字段按位计算的长度，默认值为32 |
| 3. | value | Y | 属性用来设置字段的值，默认值为0,本属性支持表达式，包括+、-、\*、/、（） |
| 4. | byteorder | Y | 属性标识字节排列顺序，字节排列顺序分为big大端序（先高字节后低字节）与small小端序（先低字节后高字节）两种，默认值为small |
| 5. | encoded | Y | 属性标识字段的编码方式，编码方式分为原码（primitive）、反码（inversion）和补码（complement）三种， 默认值为primitive |
| 6. | repeated | Y | 属性标识当前字段重复出现的次数， 默认值为1。本属性支持表达式，包括+、-、\*、/、（） |
| 7. | toenum | Y | 属性枚举转换属性,将字段转换成对应的枚举类型 |
| 8. | min | N(规则) | 接收数据时，字段值小于min值抛出异常 |
| 9. | max | N(规则) | 接收数据时，字段值大于max值抛出异常 |
| 10. | check | N(校验) | 校验算法  举例  **integer** che1 check=sum8; |
| 11. | checkrange | N(校验) | 校验数据范围  举例  **integer** che2 check=sum8checkrange=(data,data3); |

* real字段

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 标识 | 是否属性 | 描述 |
| 1. | isdouble | Y | 属性标识浮点数的是否为双精度（double），isdouble配置为false时表示单精度浮点数（float），默认值为false。 |
| 2. | byteorder | Y | 属性标识字节排列顺序，字节排列顺序分为大端序（先高字节后低字节）与小端序（先低字节后高字节）两种，默认值为small |
| 3. | value | Y | 属性用来设置字段的值，默认值为0。本属性支持表达式，包括+、-、\*、/、（） |
| 4. | encoded | Y | 属性标识字段的编码方式，编码方式分为原码（primitive）、反码（inversion）和补码（complement）三种， 默认值为primitive。 |
| 5. | repeated | Y | 属性标识当前字段重复出现的次数， 默认值为1。本属性支持表达式，包括+、-、\*、/、（） |

* block字段

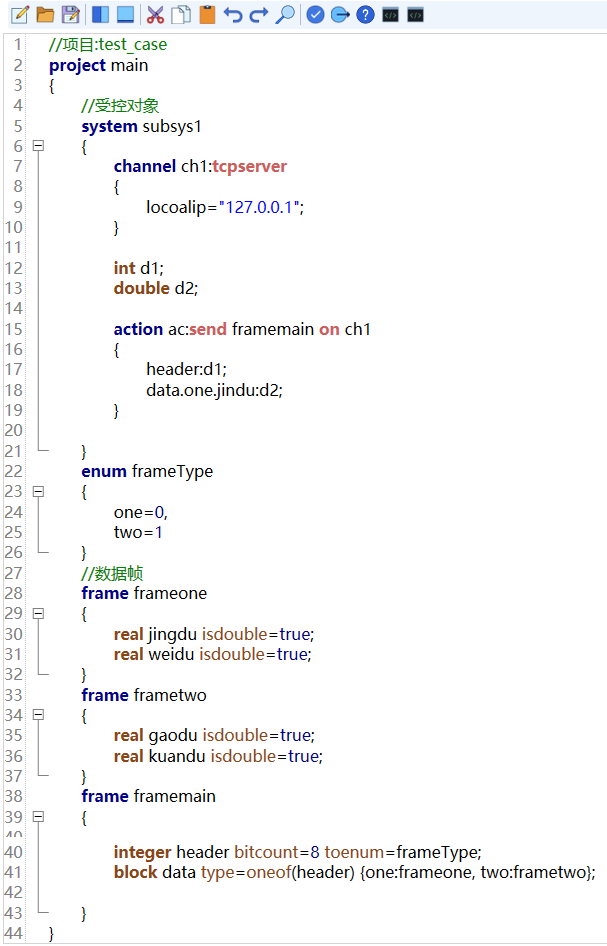
|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 标识 | 是否属性 | 描述 |
| 1. | type | Y | 标识block字段所对应的嵌套报文格式，默认值为null。可选的赋值方式为以下三种之一：  1.已经配置的数据帧名称  2.一组字段定义  3.oneof函数  代码举例：  枚举:  **enum** frameType {    one=0,    two=1 }  数据帧:  **Block**data type=oneof(header) {one:frameone, two:frametwo,other:framedefault}; |

嵌套举例：

首先新建一个项目，项目中运用嵌套的方法如下：

1. 定义一个枚举。
2. 定义主数据帧，子数据帧。主数据帧将字段应用到枚举类型上。

项目代码如下：



其中：framemain就是主数据帧，包括header字段，toenum属性header字段转换成到枚举类型。data为数块字段，type对应嵌套类型的报文格式，oneof函数枚举header字段，枚举匹配one值，data就为frameone子数据帧，匹配two值，data就为frametwo子数据帧。在action动作中给数据赋值时就用data.one.jingdu逐级取到子数据帧的值。

#### 枚举项配置

枚举配置项与C#、C++中枚举类型定义方式一致，主要用于辅助完成分系统与数据帧的配置。

## 生成代码集成方法

将系统生成的\*.cs代码文件插入大系统中，同时在大系统中引用FrameIOInterface.dll、FrameIORun.dll、FrameIODriver.dll即可完成生成代码集成。

## 软件配置使用示例

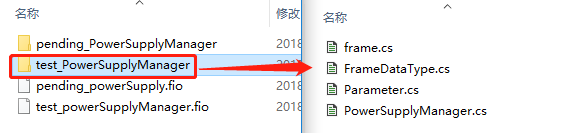
系统的使用主要包括两部份内容，一个是配置系统所用的测试项目、分系统、分系统属性、分系统通道、分系统执行动作及完成通信的数据帧格式信息，用编辑好的配置信息生成系统运行时的源代码信息。一个是将系统生成的源代码文件加入到项目工程文件中。

下面以一个TCP服务端与客户端通信示例讲述系统的应用方法。主要的操作为新建两个项目，一个项目配置TCP的Server端，一个项目配置TCP的Client端。

操作分成两部分，配置生成源代码和将源代码加入工程文件。

（1）配置生成源代码

* 用FrameIO软件新建项目用于TCP服务器端。项目名称项目中配置分系统信息、分系统属性信息、通道信息、数据帧信息。项目文件名称为test\_powerSupplyManager.fio。点击“生成”菜单按钮，在项目文件目录下生成一个与项目文件同名的文件夹，里面包含生成的源代码。
* 同样用软件新建一个TCP客户端项目，配置相关信息。项目名称文件pending\_powerSupply.fio点击“生成”菜单按钮，在项目文件目录下生成一个与项目文件同名的文件夹，里面包含生成的源代码。

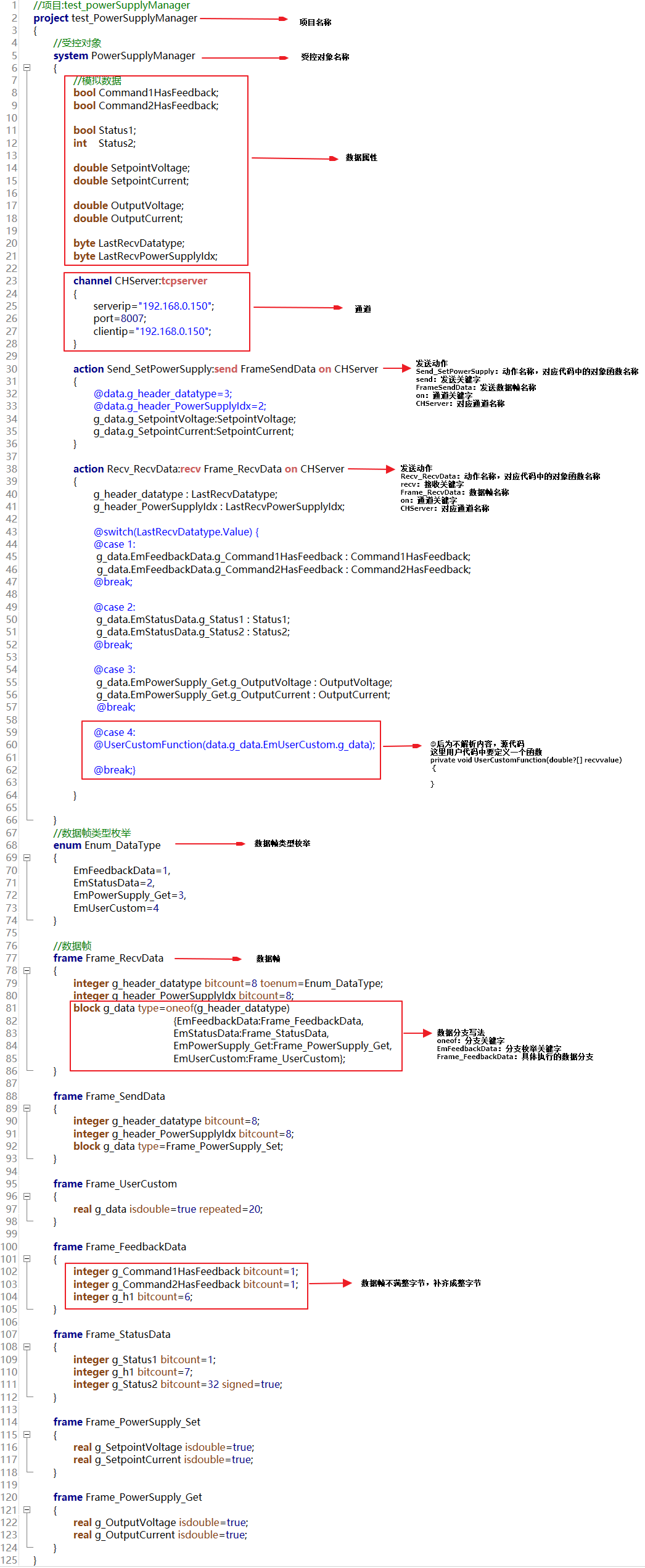


（2）将源代码加入工程

* 工程中引用FrameIODriver.dll、FrameIOInterface.dll、FrameIORuntime.dll三个动态库文件，将生成的源代码文件拷贝到工程文件目录中编译执行就可以了。

（3）项目文件的内容及生成的代码内容。

* test\_powerSupplyManager.fio文件内容

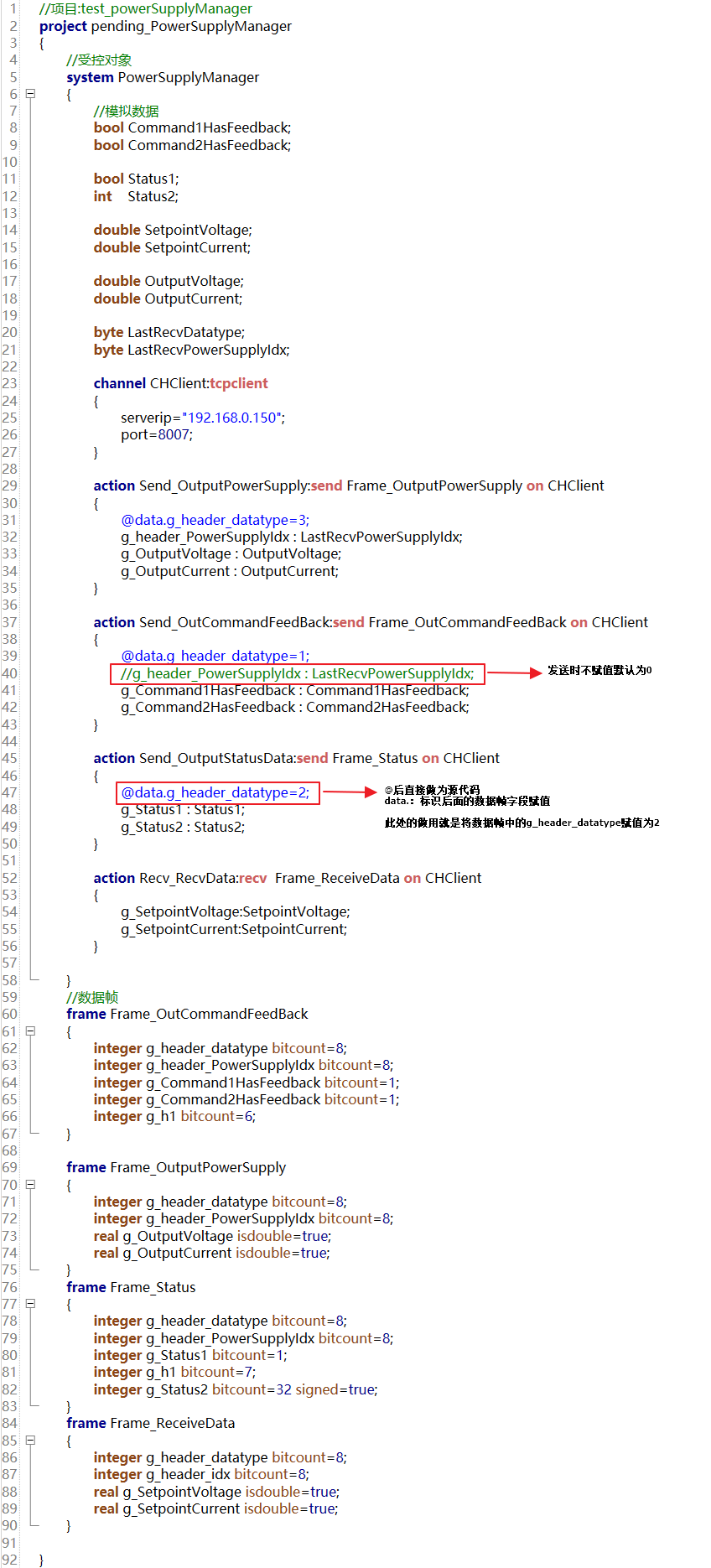


**项目文件结构示意图**

项目的主体结构为以上图所示，项目具体内容如下：

|  |  |
| --- | --- |
| 名称 | 内容 |
| * 1个项目 | * 1个受控对象PowerSupplyManager * 10个属性 * 1个发送动作 * 1个接收动作 * 1个数据帧类型枚举Enum\_DataType * 8个数据帧 |
| 受控对象PowerSupplyManager | |
| * 10个模拟数据 |  |
| * 1个通道 |  |
| * 1个发送动作 |  |
| * 1个接收动作 |  |
| 数据帧枚举 | |
| Enum\_DataType |  |
| 数据帧 | |
| Frame\_RecvData |  |
| Frame\_SendData |  |
| Frame\_UserCustom |  |
| Frame\_FeedbackData |  |
| Frame\_StatusData |  |
| Frame\_PowerSupply\_Set |  |
| Frame\_PowerSupply\_Get |  |
| 注意 1.接收动作中switch分支的写法 | |

* pending\_powerSupply.fio文件内容



项目的主体结构为以上图所示，项目具体内容如下：

|  |  |
| --- | --- |
| 名称 | 内容 |
| * 1个项目 | * 1个受控对象PowerSupplyManager * 1个数据帧类型枚举Enum\_DataType * 4个发送动作 * 1个接收动作 * 4个数据帧 |
| 受控对象PowerSupplyManager | |
| * 10个模拟数据 |  |
| * 1个通道 |  |
| * 3个发送动作 |  |
| * 1个接收动作 |  |
| 数据帧枚举 | |
| Enum\_DataType |  |
| 数据帧 | |
| Frame\_RecvData |  |
| Frame\_SendData |  |
| Frame\_UserCustom |  |
| Frame\_FeedbackData |  |
| Frame\_StatusData |  |
| Frame\_PowerSupply\_Set |  |
| Frame\_PowerSupply\_Get |  |
|  |  |

1. 新建项目

运行FrameIO.exe软件，点击“新建项目”按钮，输入项目名称（test\_com.flo）,系统新创建一个项目配置文件。

1. 配置相关信息。

配置相关信息主要包括配置通道信息、配置数据帧信息、配置action动作信息。

a)配置通道信息

创建一个新的项目文件后，在代码编辑区域配置通道信息，串口号、波特率、校验位、数据位、停止位。

b)配置数据帧信息

在数据帧(Frame)区块,输入相应数据帧字段内容。

1. 配置action动作

配置动作的标识、发送的数据帧及所用的通道和数据内容。

示例的配置代码如下所示：

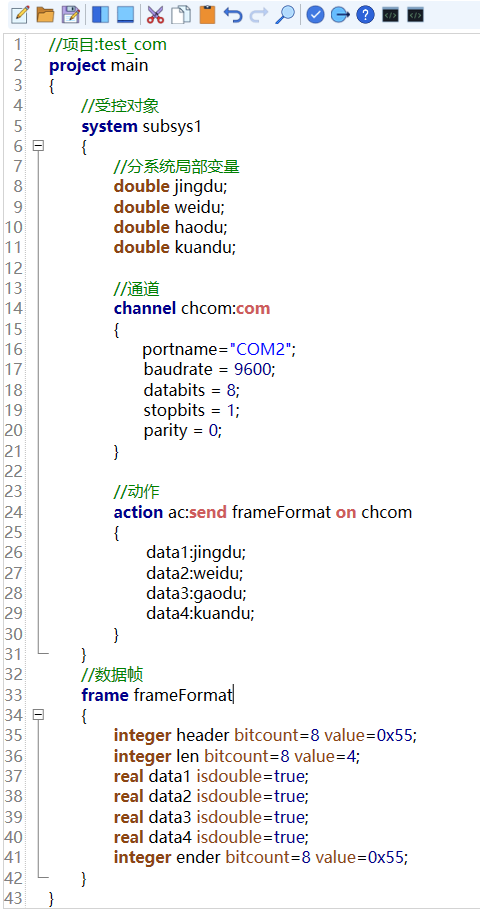


图5 配置文件编辑示例

以上示例新建了一个项目文件，项目文件上配置了一个串口通道，通道配置属性为通道号COM2、波特率9600、无校验、8位数据位、1位停止位。

数据帧格式配置为

|  |  |
| --- | --- |
| 数据字段 | 数据字段描述 |
| header | 帧头：8位整数 |
| len | 数据长度：8位整数 |
| data1 | 数据1字段：双精度数据 |
| data2 | 数据2字段：双精度数据 |
| data3 | 数据1字段：双精度数据 |
| data4 | 数据1字段：双精度数据 |
| ender | 帧尾：8位整数 |

执行动作为在串口通道（COM2）上发送数据帧（FrameFormat）。

1. 生成代码

用以上生成的配置文件，点击“生成代码”按钮，自动生成可运用到大系统中的.cs源代码文件。文件结构及内容如下：

//TODO

# 相关文件