**通信接口及协议配置组件设计方案**

# 引言

## 应用背景

在业务系统开发过程中经常需要处理类型定义、接口通信、协议解析等重复而繁琐的开发工作，这些内容处理起来不仅费时，而且经常容易出错，修改维护也比较困难。通信接口及协议配置组件正是针对这一难题，将繁复的开发任务进行自动化处理；只需输入少量配置信息便可自动生成指定的通信代码库，通信代码库将受控对象的类型定义、接口驱动调用、通信报文格式解析等技术细节进行内部封装，对外提供一组简单统一的API接口供业务系统开发时使用。

通信接口及协议配置组件（以下简称“配置系统”）是一套专用于实现快速创建软件系统底层通信模块的工具软件，配置系统主要用来实现：创建配置信息，并根据创建的配置信息自动生成通信代码库，通信代码库将数据输入输出操作封装成受控对象的方法提供给业务系统使用。在开发业务系统时，开发人员只需直接使用通信代码库中的API接口（受控对象的方法）即可实现完整的数据通信功能，而无需直接处理通信协议的底层细节。使用配置系统即省去许多繁复的开发工作，又增强了业务系统的健壮性与可维护性，正真使开发人员更集中精力去完成应用系统的业务逻辑。

## 定义

* 配置系统

配置系统是指实现配置信息编辑、解析，及自动生成通信代码库功能的软件系统的统称。配置系统的输入内容是配置信息，输出内容是通信代码库。

* 配置项

配置项指配置信息中的条目，是按照自定义格式书写的文本代码，每个项目的配置信息均由一系列配置项组成，每一配置项均与面向对象中的域对应。

* 通信代码库

通信代码库是配置系统的输出内容，是一组C#或C++源代码，对外有统一的API可供访问，内部实现了完整的通信要素管理。

* 受控对象

受控对象用以指代业务系统中的功能组件，是实现独立功能的分系统；受控对象拥有自己的属性，并使用硬件接口资源进行数据报文收发。受控对象是配置系统中的核心处理对象。

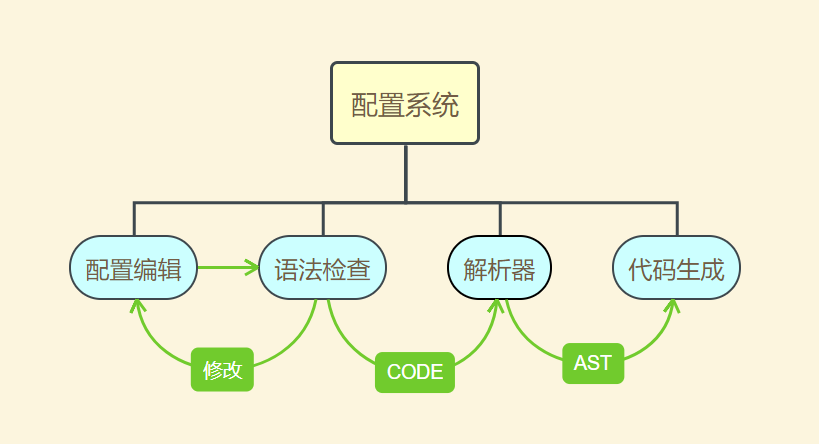
* 数据帧

数据帧是按照报文格式排列的一段数据内容，是受控对象进行数据通信时的最小数据单位。

# 总体架构

## 功能模块

配置系统由四个主要功能模块组成，分别为：配置编辑模块，语法检查模块、解析器和代码生成器。



配置系统功能模块

* 配置编辑模块

配置编辑模块用于创建配置信息，创建配置信息的方式分为可视化编辑和代码编辑两种可选模式，两种模式可以自由切换。

可视化编辑模式下，用户通过选择图形界面上的菜单、工具栏按钮来生成受控对象的各个域（属性、方法和事件）；对每个域内部的详细定义，通过UI表单的方式实现编辑功能。在可视化编辑模式下，用户在图形界面上的每一步操作均会由对应的配置项代码产生，配置信息由配置系统自动管理，用户只需按照图形界面上的指引填入相应内容即可生成完整的配置信息，非常适合初级用户使用。

代码编辑模式下，用户的主工作界面是一个WYSIWYG代码编辑器，使用代码编辑器可直接编辑配置信息的原始文本。每个项目的配置信息都是一个独立的配置文件，配置文件由一系列自定义配置项代码组成。代码编辑器实现代码输入、语法高亮显示、复制、粘贴、剪切、撤销、恢复等完善的WYSIWYG编辑功能，对具有一定使用经验的用户，代码编辑模式可大幅提高工作效率。

* 语法检查模块

语法检查模式用来对配置项代码执行语法审核，找出不符合语法规则的配置项代码，并提示给用户，以便进一步修改更正。

配置项代码采用自定义语法规则，在借鉴一些最常用的程序语言规则基础上，针对配置系统进行专门的优化设计，语法简洁明了；生成的配置代码文件可读性高，非常易于修改维护和重复使用。

语法检查模块不仅能准确找到配置项代码中的语法错误，而且还可以输出非常直观的提示信息，能准确提示出现错误的具体位置（行号和列号），在代码编辑器中对出现语法错误的关键词句进行明显的可视化标示。

* 解析器

解析器将文本格式的配置信息转换为格式化的抽象语法树（AST），从而实现对配置项代码的解释与翻译工作。

通过语法检查后的配置项代码交由解析器进行处理，解析器逐句分析配置项代码的中的符号，创建符号表，解析语法与语义，按照自下而上的顺序生成抽象语法树（AST）。抽象语法树（AST）是下一步代码生成的工作基础。

* 代码生成器

代码生成器用于直接产生C#或C++格式的通信代码库。

代码生成器在得到解析器生成的符号表与抽象语法树（AST）后，按照自上而下的顺序遍历全部抽象语法树（AST），并在抽象语法树（AST）的每个节点位置执行通信代码生成任务。执行通信代码生成任务时，代码生成器首先根据当前节点的语法含义动态生成一组程序代码，然后加载C#或C++代码模板，并将动态代码插入到代码模板中。在遍历完全部的抽象语法树（AST）后，代码生成器重新整理生成的代码文件，按照统一的目录结构规则输出完整的通信代码库。

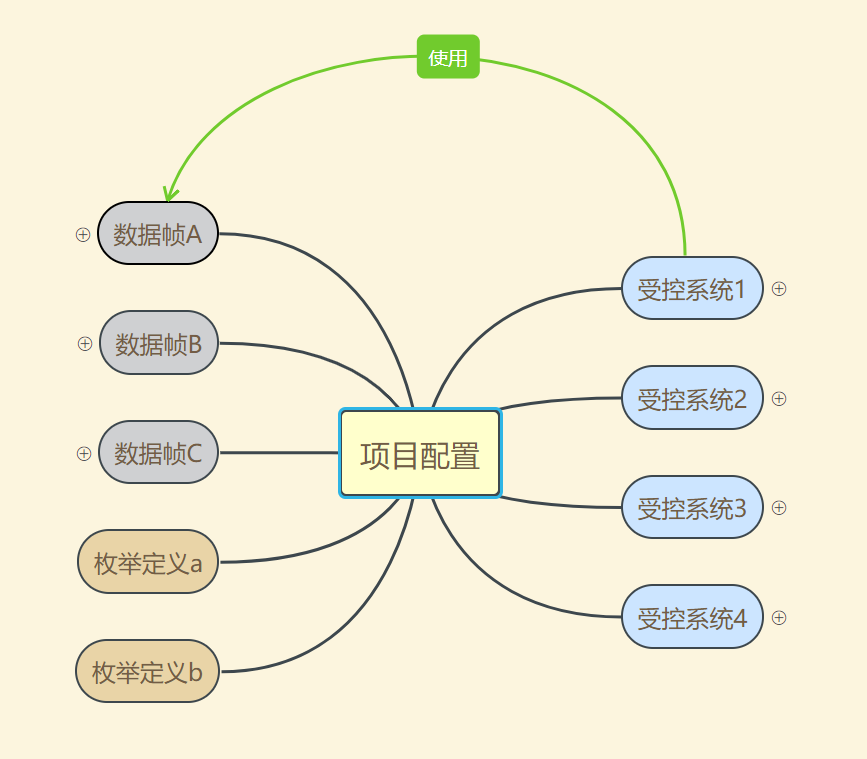
## 输入输出设计

配置系统的输入内容是配置信息，输出内容是通信代码库；这两部分内容与配置系统的使用者直接产生关系，是使用者最关心的内容，本节针对这两部分内容的总体设计进行专门说明。

* + 1. 配置信息

每个项目的配置信息都是一个文本格式的文件，由一组自定义格式的配置项代码编写而成，配置项分为三类：受控对象配置项、数据帧配置项和枚举配置项；其中受控对象配置项是配置信息的最主要内容，数据帧与枚举配置项用以辅助完成对受控对象的配置。

配置信息采用自定义格式编写，其中的关键字与语法规则均为可定制内容，可根据不同的业务特征进行匹配。



配置信息分类

* 受控对象配置项

受控对象配置项用来对受控对象的状态与IO操作进行配置。受控对象配置项按照面向对象模型进行设计，配置项内容与面向对象模型的对应关系如下：

* 每个受控对象对应一个独立的分系统类
* 受控对象的状态值对应分系统类的公有属性，状态值数据类型为值类型或值类型数组
* 受控对象通信时使用的硬件通道对应分系统类的私有成员，使用者无需直接操作这些硬件通道
* 受控对象产生的IO操作对应分系统类的公有方法，对IO操作进行配置时需指定每项IO操作的方向、所使用的硬件通道和数据帧
* 数据帧配置项

数据帧配置项用于配置受控对象通信时使用的报文格式，每一种报文格式对应一个数据帧配置项。

数据帧配置项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。数据帧的每个字段可按照数据类型或按位进行配置，字段类型既与受控对象的状态值类型兼容，又能与已存在的各类报文格式兼容。

对数据帧字段的配置，除配置数值类型外，还可配置计算值、默认值、最大值、最小值、CRC校验规则、动态解析规则等内容。

* 枚举配置项

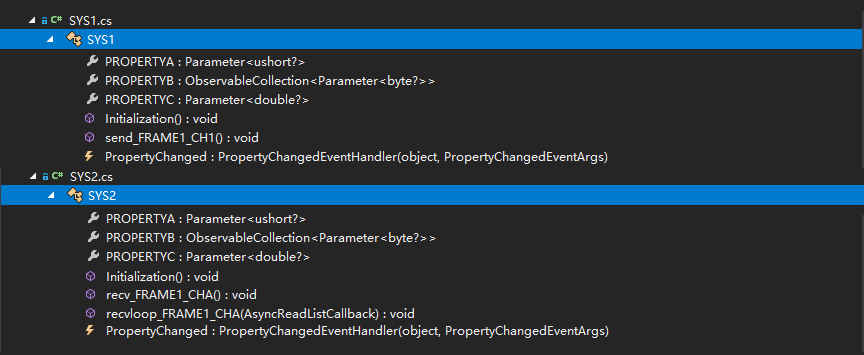
枚举配置项与通信代码库中用到的枚举定义完全一致，用以辅助完成受控对象配置与数据帧配置。

* + 1. 通信代码库API

通信代码库由配置系统自动生成，通信代码库将数据通信中需要处理的诸多要素和环节进行了隐藏封装，仅公开一组简单的API供业务系统使用；通信代码库按照面向对象模式进行设计，API的具体形式为类的公有域。

每个受控对象对应一个分系统类，受控对象的状态值对应分系统类的属性，输入输出操作对应分系统类的公有方法，当分系统类的属性值发生改变时，会自动触发相应的消息事件。

通信代码库API示例如下：



通信代码库API示例

* 属性

分系统类的属性具有公用访问权限，这些属性由配置项中的受控对象状态值产生，通信代码库的使用者通过访问这些属性来获取受控对象的状态值。属性的数据类型全部采用泛型Parameter<T>，在C#代码里，T代表一个值类型的Nullable类型；值类型数组对应的属性类型为ObservableCollection<Parameter<T>>类型。

在C#中，分系统类的属性可以作为WPF控件的数据源进行绑定，方便实现数据驱动模型下的编程开发。

* 方法

分系统类的公用方法分两类：系统资源相关的方法及输入输出操作方法。

系统资源相关的方法用于初始化系统资源及资源的释放操作，如打开或关闭硬件端口，这些方法具有统一通用的名称，如Initialization()、Release()等。

输入输出操作方法由受控对象的IO操作配置项产生，根据配置项中指定的IO方向、通道及使用的数据帧自动命名，每一个IO配置项都有对应的方法。

分系统类的公用方法是通信代码库最重要的API，开发人员主要使用这些方法来完成底层接口通信，这些方法按照统一的规则进行命名，通常没有输入输出参数，非常易于使用；方法执行过程中出现错误（包括硬件错误）时，通信代码库的内部实现会自动将这些错误信息转换为异常对象，抛出给业务系统主线程。

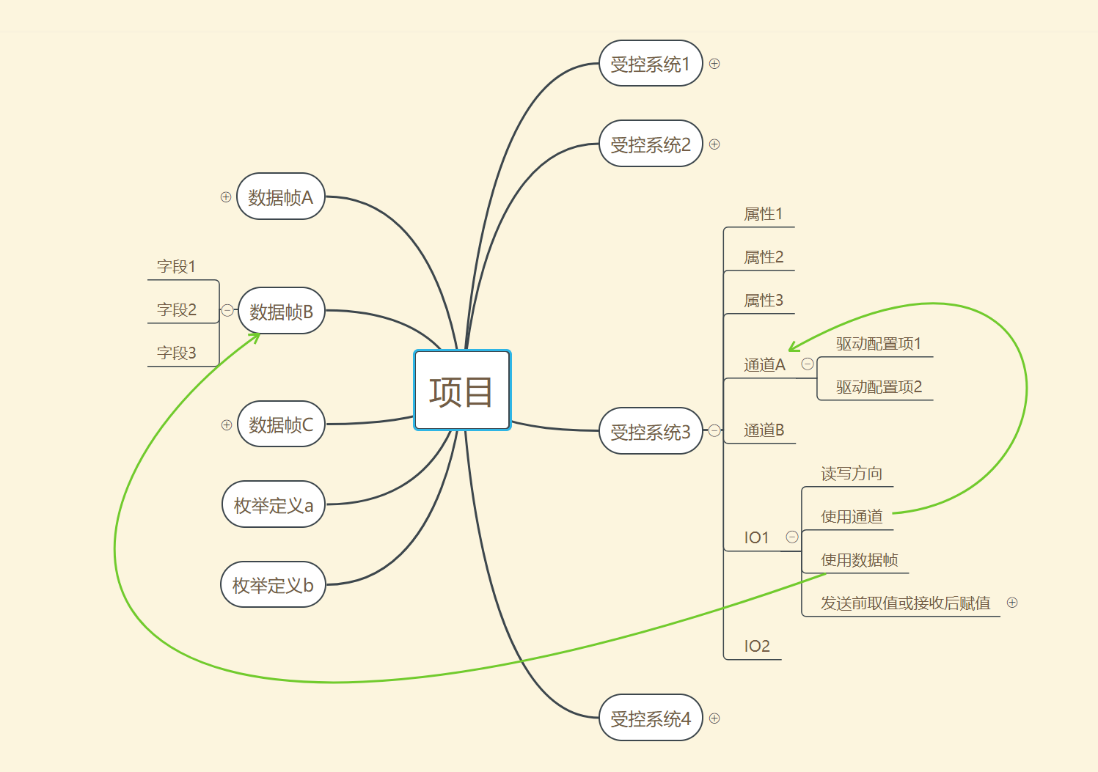
* 事件

分系统类的事件由代码生成器自动创建，在C#中该事件的类型为PropertyChanged，该事件在分系统类的公有属性值发生改变时自动触发。通过在分系统类的对象上订阅该事件，便可实现对受控对象状态值的动态监控。

# 详细设计

## 配置项内容设计

配置内容分为受控对象配置项、数据帧配置项和枚举配置项三类，其中核心的配置项是受控对象配置项，数据帧与枚举配置项用以辅助对受控对象进行配置，配置项的组成关系如下图：



配置项结构图

以下对各类配置项进行逐一说明：

* + 1. 受控对象配置项

受控对象配置项主要描述三方面内容：受控对象有哪些状态值需要管理，受控对象使用什么硬件接口进行通信，以及通信时按照什么样的报文格式收发数据。

* 受控对象属性

受控对象配置项中，每项属性代表一项需要管理的受控对象状态值。向受控对象配置项中增加属性配置时，需要指定属性名称和数值类型。

属性的名称为一个字符串标识，以字母打头，后跟若干个下划线、字母或数字的组合，配置系统使用者需确保该名称标识与业务系统中的标识没有冲突。

数值类型为以下选项之一：bool、byte、sbyte、ushort、short、uint、int、ulong、long、float、double或以上类型的数组。对于数组类型还需配置数组的长度，数组长度可以为固定的整数值或运行时动态计算的值。

* 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、串口、TCPServer、TCPClient、UDPServer、UDPClient、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。

**CAN类型通道的参数：**

**串口类型通道的参数：**

**TCPServer通道的参数：**

**TCPClient通道的参数**

**UDPServer通道的参数**

**UDPClient通道的参数**

**DI通道的参数**

**DO通道的参数**

* IO 操作

IO操作是受控对象进行数据传输的动作命令，每项IO操作配置代表受控对象进行一次数据输入输出时执行的具体动作。向受控对象配置项中增加IO操作配置时，需要指定IO操作的方向、所使用的通道、使用的数据帧（即具体的报文格式）以及相关赋值取值动作。

操作方向必须配置为发送或接收其中之一，分别代表数据输出和输入数据。

IO操作使用的通道只能引用当前受控对象中已配置通道的名称。

IO操作使用的数据帧（即具体的报文格式）只能引用当前项目中已配置数据帧的名称。

取值动作是指在发送数据前，获取受控对象的属性值，并传递给数据帧的对应字段；赋值动作是指在接收完数据帧后，将完成解析的数据帧字段值赋值给受控对象属性。进行赋值取值动作配置时需要指定受控对象属性与数据帧字段间的对应关系。

* + 1. 数据帧配置项

数据帧配置项用以配置传输协议中规定的报文数据格式，数据帧配置项是受控对象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。

* 字段类型与属性

数据帧中的字段类型有：int、real、string、block四种；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种最常用的选择方式。

**int字段属性**包括：singed、bitcount、default 、byteorder、encoded、align等。

signed属性标识整数是否有符号

bitcount属性标识字段按位计算的长度

default属性用来设置字段的默认值

byteorder属性标识字节排列顺序

encoded属性标识字段的编码方式，align属性标识无符号整数的对齐方式。

repeated属性标识当前字段重复出现的次数

**real字段属性**包括：precision、byteorder、encoded等。

precision属性标识浮点数的精度

byteorder属性标识字节排列顺序

encoded属性标识字段的编码方式

repeated属性标识当前字段重复出现的次数

**string字段属性**包括：tail、aligned

tail属性标识字符串的结尾标志

alignedbytes属性标识字符串进行整字节对齐时的字节长度

repeated属性标识当前字段重复出现的次数

**block字段属性**包括：type、bytesize

type属性标识block字段所对应的嵌套报文格式

bytesize属性标识当前字段的字节长度

repeated属性标识当前字段重复出现的次数

* 字段操作

字段操作用于对协议解析中的数值转换、计算、动态解析、验证规则等操作进行配置。

大部分解析过程中的数值转换均由解析器自动处理，需要显式配置的数值转换操作为toenum，用于执行将字段值转换为枚举类型的值。

计算操作用于运行时执行表达式计算，表达式是由加、减、乘、除、括号、字段名称组成的计算公式。

动态解析用于执行运行时动态选择报文解析方法的操作，如根据某一字段的值对另一个字段选择不同的解析方式。

验证规则用于判断字段的值是否符合一定的规则，其中包括：max、min、crc，分别用于配置最大值、最小值、校验规则。

* + 1. 枚举配置项

枚举配置项与C#、C++中枚举类型定义方式一致，主要用于辅助完成受控对象与数据帧的配置。

## 配置项编辑方式

编辑配置项的方式分为可视化编辑和代码编辑两种可选模式，两种模式可以自由切换。

* + 1. 可视化编辑模式
    2. 代码编辑模式

## 配置项处理

在生成完整的配置项后，配置系统便可以进行配置项处理，配置项处理由两个主要环节组成：语法检查与配置项解析。

* + 1. 语法检查
    2. 配置项解析

## 生成通信代码库

* + 1. 生成分系统类代码
    2. 生成数据帧类代码

# 系统实现

## 开发与运行环境

## 关键技术路线

## 质量控制

基于成熟类库解析实现

异常处理机制

# 关于凯云

# 实操示例