

# 通信接口及协议配置组件

## 技术实现方案

凯云联创（北京）科技有限公司

2018 年 9 月

## 目录

一、技术响应偏离表.....	1
二、服务响应偏离表.....	4
三、通信接口及协议配置组件技术方案.....	6
1. 引言.....	6
1.1. 应用背景.....	6
1.2. 定义.....	6
2. 总体架构.....	7
2.1. 设计原则.....	7
2.2. 功能模块.....	8
2.3. 输入输出设计.....	10
2.3.1. 配置文件.....	10
2.3.2. 生成代码.....	12
3. 方案设计.....	13
3.1. 通信模块设计.....	13
3.2. 配置项内容设计.....	14
3.2.1. 受控对象配置项.....	15
3.2.2. 数据帧配置项.....	17
3.2.3. 枚举配置项.....	20
3.3. 配置项编辑方式.....	20
3.3.1. 可视化编辑模式.....	21
3.3.2. 代码编辑模式.....	21
3.4. 配置项处理.....	22
3.4.1. 语法检查.....	22
3.4.2. 配置项解析.....	23
3.5. 代码生成.....	23
3.5.1. 分系统类代码.....	24
3.5.2. 数据帧类代码.....	24
3.5.3. 枚举定义代码.....	25
3.6. 异常处理.....	26
4. 系统实现.....	26
4.1. 开发与运行环境.....	26

4.2. 关键技术路线.....	26
4.3. 技术风险分析及应对措施.....	27
5. 关于凯云科技.....	27
6. 同类项目执行情况.....	28
7. 安装、调试、验收方法及标准.....	28
8. 质量保证.....	29
9. 培训、售后服务承诺.....	29
10. 交付物.....	29
11. 交货期.....	30
四、附件.....	31

# 一、技术响应偏离表

## 技术响应偏离表

采购项目名称：通信接口及协议配置组件

序号	采购要求	竞标应答	差异说明
1.1	提供独立的底层通信接口模块，包括TCP（服务器和客户端）、UDP、串口、CAN和DI/DO。其中CAN需提供对广州致远和研华两个厂家设备的支持，DI/DO需提供对研华厂家设备的支持；	通信模块提供独立的底层通信接口，实现对硬件通道进行输入输出操作的调用。通信接口类型包括TCP（服务器和客户端）、UDP、串口、CAN和DI/DO。其中CAN通信接口实现对广州致远和研华两个厂家设备驱动的封装，DI/DO接口实现对研华厂家设备驱动的封装。	无差异
1.2	提供协议配置模块，可对TCP、UDP、串口、CAN和DI/DO这些通信方式进行协议配置。支持配置任意格式的通信协议。数据内容部分，支持配置常用的数据格式，包括bool、byte、sbyte、ushort、short、uint、int、ulong、long、float、double、枚举类型以及数组类型。支持按位或任意位的组合（不大于8位）进行数据定义与解析。支持通过配置公式的方式进行数据自动计算。支持数据校验算法及其配置选取（单、双字节和校验、CRC校验以及其它自定义校验等）。	协议配置模块用于对协议通信中的通道、通信主体、数据帧等内容进行配置。通道配置用于设置通信方式及初始化通信环境的各项参数，通信方式包括TCP、UDP、串口、CAN和DI/DO等；通信主体配置用于设置数据内容，支持的数据格式包括bool、byte、sbyte、ushort、short、uint、int、ulong、long、float、double、枚举类型以及数组类型；数据帧配置用于设置通信报文的数据格式及解析方式，数据帧配置支持按位或任意位的组合（不大于8位）进行数据定义与解析，支持通过配置公式的方式进行数据自动计算，内置常用数据校验算法（单、双字节和校验、CRC校验等）供配置时选取，同时支持其它自定义校验。	无差异
1.3	协议配置数据以文件方式存储，格式需要采用Xml或JSON或其他自定义文本格式。文件以项目为单位，一个项目一组独立的文件，一个项目中包含多个受控对象，与受控对象之间的通信方式（TCP、UDP、串口等）可配置，每种通信方式的通信参数可配置，比如串口和CAN的波特率等；	每个项目的配置信息都是一个自定义格式的文本文件，文件内容由三类配置项组成：受控对象配置项、数据帧配置项和枚举配置项，每类配置项由多个配置对象组成。受控对象作为通信主体，可配置各自的属性、通信通道（包括通信方式、通信参数）、通信数据帧等内容。	无差异
1.4	提供独立的协议配置软件，支持以面向对象的方式进行协议配置（即不是基于数据包本身来进行协议配置，而是基于受控对象进行协议配置，包括受控对象的属性、收发数据包、通信方式及其属性等），从而实现对协议	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： 1. 每个受控对象对应一个独立的分系统类； 2. 受控对象的状态值对应分系统类的公有属性； 3. 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； 4. 受控对象产生的IO操作对	无差异

	<p>配置文件的新建、编辑和删除操作。协议配置软件支持可视化的配置方式，同时支持对协议配置文件以文本方式直接编辑（提供语法检查等功能）</p>	<p>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格式等参数内容。</p> <p>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，两种模式均以面向对象的方式实现对协议配置文件的新建、编辑和删除操作。通信配置系统包含完整的语法检查模块，该模块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，并以直观的方式提示给用户，以便进一步修改更正。</p>	
1.5	<p>提供针对协议配置的代码自动生成软件（可与协议配置软件合并）。选择某个项目协议配置文件后，可选择其中的一个或多个受控对象自动生成代码，生成的代码直接体现为受控对象本身（包括受控对象的属性、收发数据包、通信方式及其属性等）。生成代码的模板允许编辑。</p> <p>生成 C# 代码时，根据项目 ID 和受控对象 ID 自动生成命名空间，每个受控对象的类文件以文件夹形式分组排列。所有的值类型变量，需要将其封装为类使用，类名为 <code>Parameter&lt;T&gt;</code>，数值使用 <code>T Value</code> 属性访问。值类型变量，泛型传入的类型均使用 <code>Nullable</code> 类型。</p> <p><code>Parameter&lt;T&gt;</code> 提供一些基本的属性，比如 <code>string Name</code>，填入中文描述。还需提供 <code>ValueChanged</code> 事件。</p>	<p>代码生成器用于直接产生 C# 或 C++ 格式的程序代码。选择某个项目协议配置文件后，可选择其中的一个或多个受控对象自动生成代码，每个受控对象对应一个面向对象的类，该类的内部域（属性、方法等）对应受控对象的属性、收发数据包、通信方式及其属性等。生成代码的格式由可编辑的代码模板设定。</p> <p>生成 C# 代码时，受控对象的类定义位于“项目 ID. 受控对象 ID”命名空间下，类文件以文件夹形式分组排列，受控对象属性数据类型为泛型类 <code>Parameter&lt;T&gt;</code>，其中 <code>T</code> 为配置项中指定的基础类型的 <code>Nullable</code> 表示。<code>Parameter&lt;T&gt;</code> 的类型值访问属性为 <code>T Value</code>，当修改 <code>Value</code> 属性值时会自动触发 <code>ValueChanged</code> 事件。除 <code>Value</code> 属性外，<code>Parameter&lt;T&gt;</code> 的还包括 <code>string Name</code>，<code>bool IsNull</code> 等基本属性。</p>	无差异
1.6	<p>应针对组件处理异常设计异常处理机制，并向上层提供异常代码传输接口。</p>	<p>当出现运行时错误时，协议配置系统首先会收集详细的错误信息，然后以抛出异常的方式向主程序报告错误，在抛出的异常对象中，包含完整的原始错误信息（错误类型、错误提示信息、报错位置、原始错误码等）。</p>	无差异
1.7	<p>开放底层硬件驱动调用接口，提供接入新的接口通信方式和硬件驱动功能。</p>	<p>通信模块将硬件设备的驱动接口或操作系统提供的通信接口封装成统一的输入输出 API，从而实现不同的通信方式和通信参数配置可以使用相同的 API 进行协议通信。其中硬件驱动调用接口封装使用凯云公司自研的 <code>FrameIO</code> 模块实现，<code>FrameIO</code> 经过多次迭代升级与多年实践验证，成熟稳定、通用性强，<code>FrameIO</code> 采用开放式架构，具备完善的扩展接口与开发文档，非常</p>	无差异

		易于扩展新的通信方式。	
--	--	-------------	--

供应商：凯云联创（北京）科技有限公司

法人授权代表：

（供应商公章） （签字或盖章）

2018 年 9 月 25 日

注：1、本表即为对本项目所列技术要求进行比较和响应；

2、该表必须按照竞争性谈判要求逐条如实填写，根据竞标情况在“差异说明”项填写正偏离或负偏离及原因，完全符合的填写“无差异”；

3、该表可扩展，并逐页签字或盖章；

4、可附相关技术支撑材料。（格式自定）

## 二、服务响应偏离表

### 服务响应偏离表

采购项目名称：通信接口及协议配置组件

序号	偏离名称	谈判项目需求	竞标应答	偏离说明
2	开发与运行环境要求	该软件组件在 Windows 操作系统下使用（Windows7 及以上版本）。 需支持 C#语言（运行库为 .NET Framework4.5 及以上版本），以及 C++语言（包括 Visual C++和 Qt）。其中对 C#语言的开发采用 Visual Studio 2017。	通信配置组件的开发环境基于 windows 操作系统(Windows7 及以上版本)，开发工具采用 Visual Studio 2017。（其中协议配置模块采用 C# WPF 开发，语法检查与解析器为 C++开发的动态链接库，代码生成器为 C#开发的 .Net 类库）。其中 .Net Framework 使用 4.5 版本，同时支持更高版本。生成代码支持 C#、C++，C#代码运行库为 .NET Framework4.5 及以上版本，支持 Visual Studio 2017；C++代码支持 Visual C++和 Qt。	无差异
3	质量要求	该软件组件开发应严格按照软件工程化要求进行，完成需求分析、软件设计、软件配置项测试等相关工作，形成软件需求规格说明、软件设计说明、软件测试说明、软件测试报告和软件用户手册，所有文件应经正式评审/会签。	本软件组件开发将严格按照软件工程化标准进行，开发组织按以下步骤迭代施行：a. 需求分析；b. 概要设计；c. 详细设计；d. 软件实现；e. 单元测试 f. 配置项测试。以上工作形成的阶段性交付物（包括需求规格说明、软件设计说明、软件测试说明、软件测试报告和软件用户手册）均需经过正式评审/会签。	无差异
		该软件组件需进行第三方测评，并提供正式的第三方测评报告。	全部阶段工作完成后，将聘请具有 CNAS 资质的第三方测评机构进行评测，并提供合格有效的测评报告。	无差异
4	培训要求	厂家应向用户提供软件组建使用相关技术培训，并提供完整的技术资料，确保用户可正常使用该组件。提供不少于 1 年的技术支持服务。	本公司就交付的软件组件提供完整技术文档（纸质 2 份、电子版 1 份）并按用户要求提供多次现场培训，确保用户正常使用全部功能。 自项目验收之日起，提供 3 年技术支持服务，服务内容包括电话咨询、远程协助、上门技术支持等，提供全年 7*24 小时远程支持服务，严重问题可按用户要求 2 小时内到达现场。	无差异
5	交付要求	厂家应向用户提供如下	本项目最终交付物包括：	无差异

		交付物： 通信接口及协议配置组件源代码及可执行程序；用户手册；需求规格说明；软件设计说明；软件测试说明；软件测试报告；软件第三方测评报告。 注：文档均提供电子版和纸质两种交付物。	通信接口及协议配置组件源代码及可执行程序；用户手册电子版与纸质版；需求规格说明电子版与纸质版；软件设计说明电子版与纸质版；软件测试说明电子版与纸质版；软件测试报告电子版与纸质版；软件第三方测评报告电子版与纸质版。	
6	时间进度要求	研制周期：自合同签订一个月内，向用户交付基于C#语言的通信接口及协议配置组件，以及配套的软件用户手册；合同签订起两个月内，向用户交付完整的通信接口及协议配置组件，以及全套文档。	自合同签订一个月内，向用户交付基于C#语言的通信接口及协议配置组件，以及配套的软件用户手册。 合同签订起两个月内，向用户交付完整的通信接口及协议配置组件，以及全套文档。	无差异

供应商：凯云联创（北京）科技有限公司

法人授权代表：

（供应商公章） （签字或盖章）

2018 年 9 月 25 日

注：1、本表即为对本项目所列服务要求进行比较和响应；

2、该表必须按照谈判要求逐条如实填写，根据竞标情况在“差异说明”项填写正偏离或负偏离及原因，完全符合的填写“无差异”；

3、该表可扩展，并逐页签字或盖章；



# 三、通信接口及协议配置组件技术方案

## 1. 引言

### 1.1. 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；只需输入少量配置代码便可自动生成指定的程序代码，生成的程序代码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 供业务系统开发时使用，从而大幅度简化底层通信模块的开发维护工作。

通信接口及协议配置组件（以下简称“通信配置组件”）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，通信配置组件主要用来实现：提供独立的通信模块，创建配置文件并自动生成程序代码，生成的程序代码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生成代码中的 API（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。

使用通信配置组件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，正真使开发人员将更多精力集中于应用系统的业务逻辑上。

### 1.2. 定义

- 通信配置组件

通信配置组件是指实现通信模块、配置文件编辑、解析，及自动生成程序代码功能的软件系统统称。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。

- 配置项

配置项指配置文件中的条目，是按照自定义格式书写的文本代码，每个项目的配置文件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。

- 生成代码

生成代码是通信配置组件根据配置内容自动生成的程序代码，是一组 C#、C++源代码文件，生成代码实现对所配置对象的定义，并提供协议通信的 API。

- 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。受控对象是通信配置组件中的核心处理对象，是协议通信的主体对象。

- 数据帧

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

## 2. 总体架构

### 2.1. 设计原则

通信配置组件的设计原则为：稳定、易用、完备、可扩。

稳定：作为业务系统共用的底层通信模块，通信配置组件必须首先保证其自身的稳定运行，本方案基于成熟的经过实践验证的软件模块构建，这是保证系统整体运行稳定的基石。

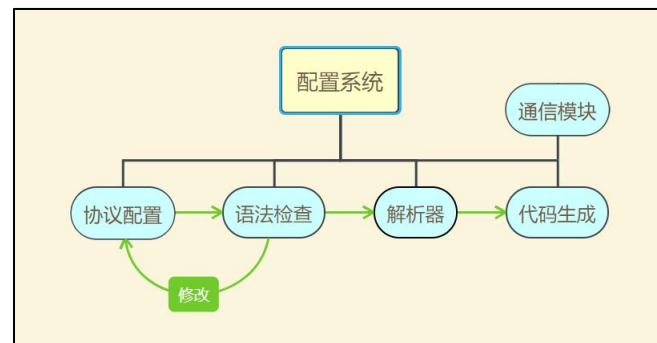
易用：作为配置型工具，通信配置组件需要易于使用，本方案提供友好的图形化操作界面，符合绝大多数用户操作习惯。

完备：作为通用型软件模块，通信配置组件需要适用于大多数协议通信场景，本方案具备完善的协议通信处置能力。

可扩：作为可重用代码库，通信配置组件需要具备可扩展的能力，本方案可以很方便的添加其它通信接口，可以与多种开发平台快速集成。

## 2.2. 功能模块

通信配置组件由五个主要功能模块组成，分别为：通信模块、协议配置模块、语法检查模块、解析器和代码生成器。



通信配置组件功能模块

### ● 通信模块

通信模块提供独立的底层通信接口，实现对硬件通道进行输入输出操作的调用。通信接口类型包括 TCP（服务器和客户端）、UDP、串口、CAN 和 DI/DO。其中 CAN 通信接口实现对广州致远和研华两个厂家设备驱动的封装，DI/DO 接口实现对研华厂家设备驱动的封装。

### ● 协议配置模块

协议配置模块用于对协议通信中的通道、通信主体、数据帧等内容进行配置。通道配置用于设置通信方式及初始化通信环境的各项参数，通信方式包括 TCP、UDP、串口、CAN 和 DI/DO 等；通信主体配置用于设置数据内容，支持的数据格式包括 bool、byte、sbyte、ushort、short、uint、int、ulong、long、float、double、枚举类型以及数组类型；数据帧配置用于设置通信报文的数据格式及解析方式，数据帧配置支持按位或任意位的组合（不大于 8 位）进行数据定义与解析，支持通过配置公式的方式进行数据自动计算，内置常用数据校验算法（单、双字节和校验、CRC 校验等）供配置时选取，同时支持其它自定义校验。

创建配置文件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）；每个域的详细配置编辑均通过 UI 表单实

现。在可视化编辑模式下，用户在图形界面上的每一步操作系统后台都会有对应的配置项代码产生，配置文件由通信配置组件自动管理，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文件，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文件的原始文本。每个项目的配置文件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脚本语言规则基础上，针对通信配置组件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提示。

- 解析器

解析器将文本格式的配置文件转换为格式化的抽象语法树（AST），从而实现了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++ 格式的程序代码。选择某个项目协议配置文件后，可选择其中的一个或多个受控对象自动生成代码，每个受控对象对应一个面向对象的类，该类的内部域（属性方法等）对应受控对象的属性、收发数据帧、通信方式及其属性等。生成代码的格式由可编辑的代码模板设定。

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程序代码生成任务。执行程序代码生成任务时，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生成代码。

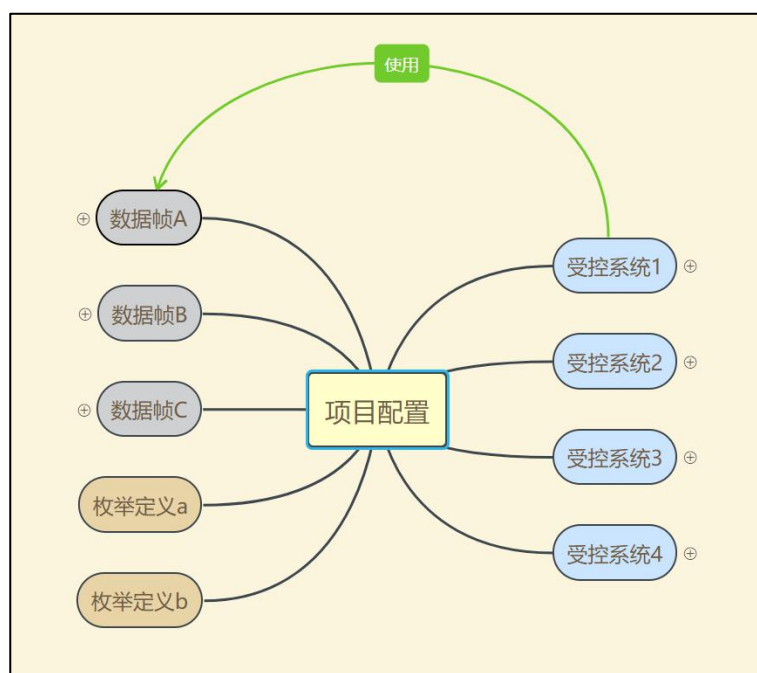
## 2.3. 输入输出设计

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；这两部分内容与通信配置组件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。

### 2.3.1. 配置文件

每个项目的配置文件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文件的最主要内容，数据帧与枚举配置项用以辅助完成对受控对象的配置。

配置文件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适配。



配置项分类

### ● 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

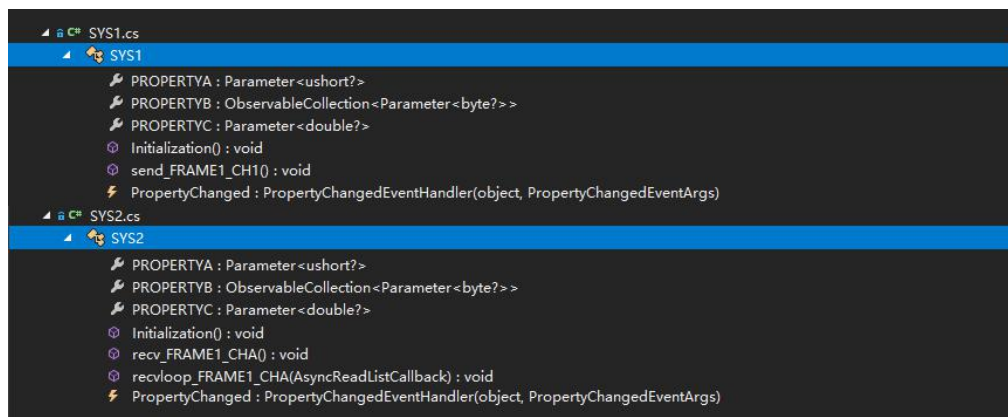
枚举配置项与生成代码中用到的枚举定义完全一致，用以辅助完成受控对象配置与数据帧配置。

### 2.3.2. 生成代码

生成代码由通信配置组件自动生成，生成代码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；生成代码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公有属性，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。

生成代码 API 示例如下：



生成代码 API 示例

- 公有属性

具有公有访问权限的分系统类属性，由配置项中的受控对象状态值产生，生成代码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 表示；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。

- 公有方法

分系统类的公有方法分两类：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生成代码最重要的 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，生成代码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。

#### ● 公有事件

分系统类的公有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。

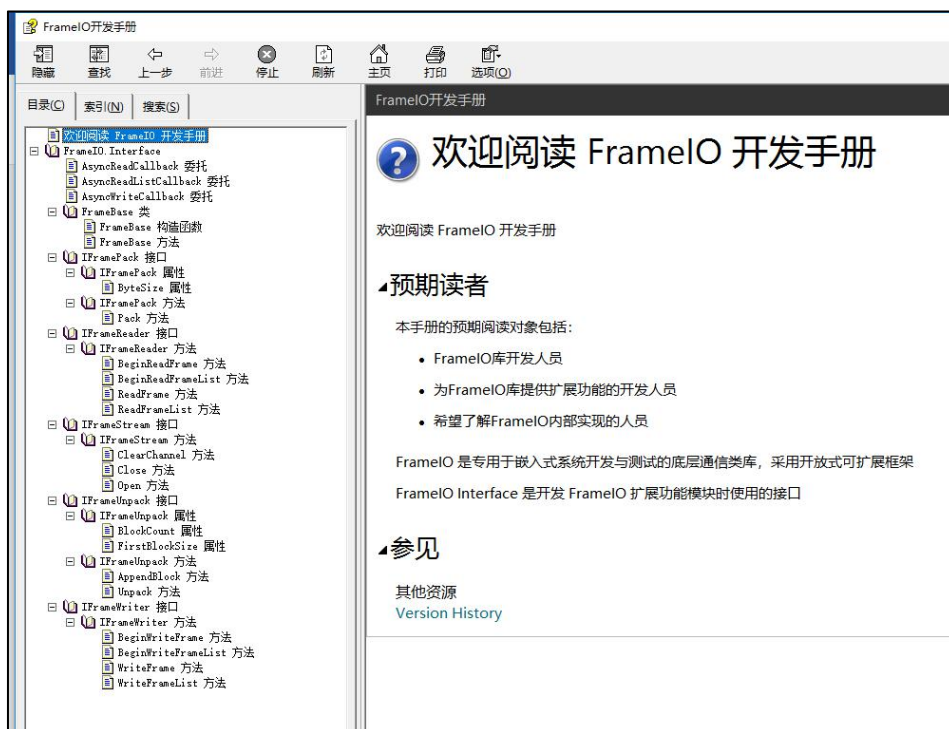
## 3. 方案设计

### 3.1. 通信模块设计

通信模块提供独立的底层通信接口，实现对硬件通道进行输入输出操作的调用。通信接口类型包括 TCP（服务器和客户端）、UDP、串口、CAN 和 DI/DO。其中 CAN 通信接口实现对广州致远和研华两个厂家设备驱动的封装，DI/DO 接口实现对研华厂家设备驱动的封装。

通信模块将硬件设备的驱动接口及操作系统提供的通信接口封装成统一的 API，从而实现不同的通信方式和通信参数配置可以使用相同的 API 进行协议通信。硬件驱动调用接口封装使用凯云公司自主研发的 FrameIO 模块实现，凯云公司基于 FrameIO 模块开发了多款嵌入式测试相关软件产品，经过多次迭代升级及多年实践验证，该模块成熟稳定、通用性强，FrameIO 具备完善扩展接口与开发文档，非常易于扩展新的通信方式。

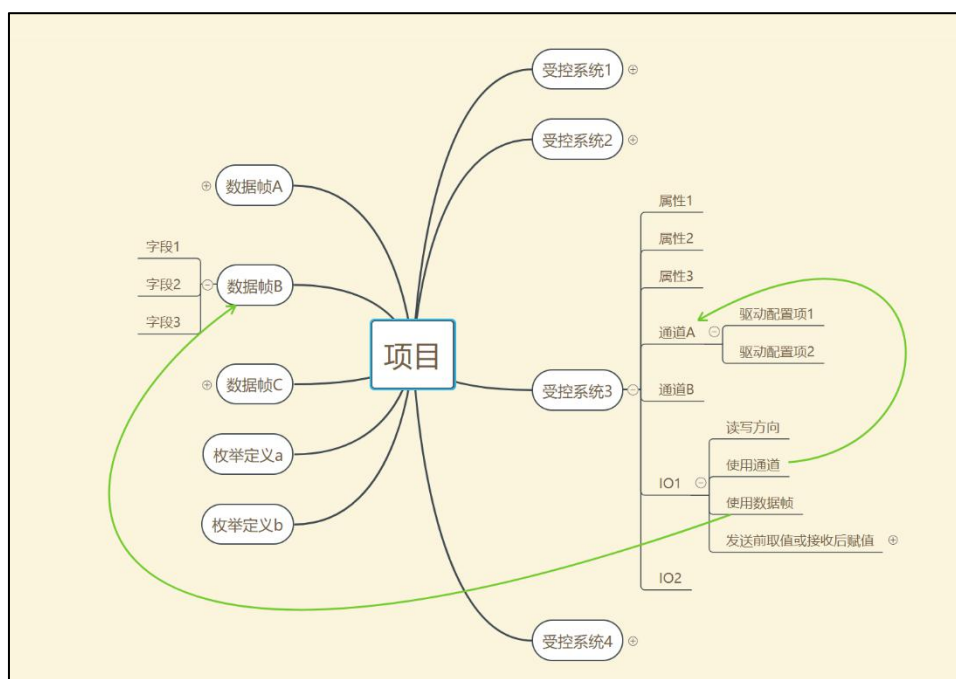




FrameIO 开发手册

### 3.2. 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：

3.2.1. 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，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。

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、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。

➤ can 类型通道的参数

序号	名称
1.	通道号
2.	波特率
3.	验收码
4.	屏蔽码
5.	工作模式(正常工作/自发自收)
6.	过滤模式(单滤波/双滤波)
7.	发送超时

➤ com 口类型通道的参数

序号	名称
1.	端口

2.	波特率
3.	奇偶校验
4.	数据位长
5.	停止位

➤ tcpserver 通道的参数

序号	名称
1.	监听 IP
2.	监听端口
3	待测设备地址
4	待测设备端口

➤ tcpclient 通道的参数

序号	名称
1.	设备 IP 地址
2.	端口
3	连接超时

➤ upd 通道的参数

序号	名称
1.	本地地址
2.	本地端口
3	远端地址
4	远端端口

➤ di 通道的参数

序号	名称
1.	通道号
2.	最小信号值

➤ do 通道的参数

序号	名称
1.	通道号

2.	最小信号值
----	-------

### ● 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 操作配置表示受控对象进行一次数据输入或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。

## 3.2.2. 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：integer、real、text、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中最常用的选择方式。

**integer 字段属性**包括：signed、bitcount、default、byteorder、encoded、

repeated 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，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。

encoded 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。

repeated 属性标识当前字段重复出现的次数。

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

isdouble 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)。

byteorder 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。

encoded 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。

repeated 属性标识当前字段重复出现的次数。

**text 字段属性**包括：tail、alignedlen、repeated 等。

tail 属性标识字符串的结尾标志。

alignedlen 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、repeated 等。

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，表达式的结果由运行时动态计算得到。

动态解析用于执行运行时动态选择报文解析方法的操作，如运行时根据某一字段的值动态选择当前字段的解析方式。

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、check，分别用于配置最大值、最小值、校验规则。其中系统内置的 check 校验算法如下表，此外 check 校验算法还支持自定义 crc 算法扩展。

内置 check 算法				
校验模式	多项式	初始值	数据位序	结果处理
sum8	8 位校验和			
xor8	8 位异或值			
sum16	16 位校验和，低字节在前，高字节在后			
sum16_false	16 位校验和，高字节在前，低字节在后			
xor16	16 位异或值，低字节在前，高字节在后			
xor16_false	16 位异或值，高字节在前，低字节在后			
sum32	32 位校验和，低字节在前，高字节在后			
sum32_false	32 位校验和，高字节在前，低字节在后			
xor32	32 位异或值，低字节在前，高字节在后			
xor32_false	32 位异或值，高字节在前，低字节在后			
crc4_itu	0x03	0x00	低位在前，高位在后	与 0x00 异或
crc5_epc	0x09	0x09	低位在后，高位在前	与 0x00 异或
crc5_itu	0x15	0x00	低位在前，高位在后	与 0x00 异或
crc5_usb	0x05	0x1F	低位在前，高位在后	与 0x1F 异或
crc6_itu	0x03	0x00	低位在前，高位在后	与 0x00 异或

crc7_mmc	0x09	0x00	低位在后，高位在前	与 0x00 异或
crc8	0x07	0x00	低位在后，高位在前	与 0x00 异或
crc8_itu	0x07	0x00	低位在后，高位在前	与 0x55 异或
crc8_rohc	0x07	0xFF	低位在前，高位在后	与 0x00 异或
crc8_maxim	0x81	0x00	低位在前，高位在后	与 0x00 异或
crc16_ibm	0x8005	0x0000	低位在前，高位在后	与 0x0000 异或
crc16_maxim	0x8005	0x0000	低位在前，高位在后	与 0xFFFF 异或
crc16_usb	0x8005	0xFFFF	低位在前，高位在后	与 0xFFFF 异或
crc16_modbus	0x8005	0xFFFF	低位在前，高位在后	与 0x0000 异或
crc16_ccitt	0x1021	0x0000	低位在前，高位在后	与 0x0000 异或
crc16_ccitt_false	0x1021	0xFFFF	低位在后，高位在前	与 0x0000 异或
crc16_x25	0x1021	0xFFFF	低位在前，高位在后	与 0xFFFF 异或
crc16_xmodem	0x1021	0x0000	低位在后，高位在前	与 0x0000 异或
crc16_dnp	0x3D65	0x0000	低位在前，高位在后	与 0xFFFF 异或
crc32	0x04C11DB7	0xFFFFFFFF	低位在前，高位在后	与 0xFFFFFFFF 异或
crc32_mpeg_2	0x04C11DB7	0xFFFFFFFF	低位在后，高位在前	与 0x00000000 异或
crc64	0x42F0E1EBA9EA3693	0x00000000 00000000	低位在后，高位在前	与 0x00000000 00000000 异或
crc64_we	0X42F0E1EBA9EA3693	0xFFFFFFFF FFFFFFFF	低位在后，高位在前	与 0xFFFFFFFF FFFFFFFF 异或

### 3.2.3. 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。

### 3.3. 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。

### 3.3.1. 可视化编辑模式

可视化编辑模式中，对所有配置项的编辑均通过 windows 系统中的标准控件进行，通过直观易操作的界面实现全部配置项的编辑功能。用户通过选择图形界面上的菜单、工具栏按钮控制配置项的添加、删除、重命名，配置项的具体内容通过 UI 表单中的输入框、复选框、下拉框等控件组合进行控制。

使用者在图形界面上的每一步操作均会自动生成相应的配置项代码，自动生成的配置项代码均为语法正确的完整代码，可视化编辑模式下不会产生任何语法错误。自动代码生成由通信配置组件后台自动执行，可视化编辑模式下使用者无需直接编辑配置代码。




配置项管理可视化界面

### 3.3.2. 代码编辑模式

代码编辑模式中，主工作界面是一个所见即所得的代码编辑器，在代码编辑器中加载完整的配置代码内容，通过代码编辑器可直接修改配置文件的原始文本。

代码编辑器除支持中英文输入，支持文本内容复制、粘贴、剪切功能，支持按修改步骤执行撤销、恢复功能。代码编辑器最左侧一列显示配置代码行号；代码行号右侧为配置代码内容，根据自定义语法规则，配置代码的不同关键词显示不同的高亮颜色，方便使用者识别，以增强配置代码的可读性。





```
1
2 //项目定义
3 project PROJECT1
4 {
5     //数据帧格式定义 可以被本项目中的所有分系统共用
6     frame @FRAME1
7     {
8         //数据帧内部字段定义
9         int @SEGMENTA bitcount=16;
10        int @SEGMENTB bitcount=8 repeated=4;
11        real @SEGMENTC;
12    }
13
14
15    //分系统定义
16    system @SYS1
17    {
18        //分系统属性定义
19        int @PROPERTYA;
20        int[] @PROPERTYB;
21        double @PROPERTYC;
22
23        //通道定义, 一个通道定义对应: 硬件板卡的一个通道 / 网络上的一个连接 / 一个串口
24        channel @CH1
25        {
26            device = can1;
27            config1 = value1;
28            config2 = value2;
29        }
30    }
```

配置代码编辑器

### 3.4. 配置项处理

在生成完整的配置项后，通信配置组件便可以进行配置项处理，配置项处理由两个主要环节：语法检查与配置项解析。

#### 3.4.1. 语法检查

配置代码语法检查的功能实现分两个步骤：首先进行符号识别，然后进行语法规义合规性判别。

符号识别采用正则表达式扫描器，对配置代码中的所有符号进行扫描识别，并记录出现的具体位置，对无法识别的符号均记录为输入错误。

语法规义判别通过BNF语法分析，对配置代码中的完整配置项进行逐项识别，识别完成后按照识别的语义进行合规性判别，对出现的不合规语法或语义冲突均记录为一项语法错误。

对整个检查过程中发现的问题均记录具体的行号和列号，并在图形界面上输出明确提示信息。

### 3.4.2. 配置项解析

配置项解析的任务由解析器完成，主要任务是将合格的配置代码转换为抽象语法树（AST），其执行过程为按照自下而上顺序构建抽象语法树（AST）。

通过语法检查后的配置项代码交由解析器进行处理，解析器逐句解析配置项代码中的符号，并创建符号表，同时按照配置项语法递归生成语法树的组成项，全部配置项解析完成后便生成了完整的抽象语法树（AST）。

配置项解析主要工作内容为：受控对象配置项解析、数据帧解析、枚举解析；其中受控对象配置项解析是抽象语法树（AST）的树根。解析时先进行数据帧和枚举解析，对数据帧中的逐个字段按照配置内容解析成对应的数据类型，并计算出数据帧打包后的字节大小，对动态计算的内容则需解析为逆波兰表达式；枚举配置项解析结果是整数值与枚举项标识对照表。

受控对象配置项解析是解析器的核心任务，对受控对象的解析需要交叉引用之前解析过程中的结果。解析受控对象配置项时，先解析属性和通道配置项，构建受控对象属性列表和通道资源列表，最后解析 IO 操作配置项，每项 IO 操作均会引用到前面各项解析结果。完成受控对象配置项解析后，便生成了完整的抽象语法树（AST）。

配置项解析是通信配置组件产生生成代码的基础，配置项解析应确保准确。该过程由通信配置组件后台自动调用，对使用者是透明的。

### 3.5. 代码生成

程序代码生成由代码生成器完成，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程序代码生成任务。执行程序代码生成任务时，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程序代码。

由通信配置组件自动生成的程序代码中包含分系统类代码、数据帧类代码和枚举定义代码，分别由配置项中的受控对象配置项、数据帧配置项和枚举配置项

对应产生。

### 3.5.1. 分系统类代码

每个受控对象配置项会生成一个分系统类定义。

生成 C# 代码时，受控对象的类定义位于“项目 ID. 受控对象 ID”命名空间下，类文件以文件夹形式分组排列，受控对象配置项中的属性配置（即受控对象的状态值）对应生成分系统类的公有属性，属性数据类型为泛型类 `Parameter<T>`，其中 `T` 为配置项中指定的基础类型的 `Nullable` 表示。例如：配置项中指定受控对象属性数据类型为 `long`，则对应生成的分系统类属性数据类型为 `Parameter<long?>`。`Parameter<T>` 的类型值属性为 `T Value`，当修改 `Value` 属性值时会自动触发 `PropertyChanged` 事件。除 `Value` 属性外，`Parameter<T>` 的还包括 `string Name`，`bool isNull` 等基本属性。如果配置项中的属性数据类型为基础数据类型的数组，对应生成的分系统类属性数据类型为 `ObservableCollection<Parameter<T>>`。`ObservableCollection` 类型实现对属性值进行修改时自动触发 `PropertyChanged` 事件。

受控对象配置项中的通道配置对应生成分系统类的通道私有变量成员，这些私有成员按照配置项进行命名并在类构造函数中进行初始化，通道私有成员仅能在分系统类内部使用，供 IO 操作实现中使用。

受控对象配置项中的 IO 操作配置对应生成分系统类的公有函数。对于数据输出操作，先将分系统类属性赋值给对应的数据帧对象，然后调用数据帧对象的打包接口获取字节流，最后将获取到的字节流在指定的通道上发送出去；对于数据输入操作，先将指定通道上读取到的字节流传递给特定数据帧对象的解包接口，完成数据帧解包后再将数据帧的字段值分别赋值给分系统对象对应的属性，如果属性值发生改变则自动触发 `PropertyChanged` 事件。

受控对象的构造函数中实现通道的初始化工作，代码实现中会根据配置项中的内容，将通道属性值传递给底层的硬件驱动程序，并按照驱动程序要求的顺序对硬件进行初始化操作。

受控对象的析构函数中实现通道资源的释放工作，代码实现中会按照驱动程序要求的顺序释放相关的硬件资源。

### 3.5.2. 数据帧类代码

数据帧配置项对应生成数据帧类代码，主要功能是报文格式解析，分系统类执行 IO 操作时需要使用数据帧类进行报文的解包与拆包，在业务系统代码中通常不会直接使用数据帧类代码。

数据帧类的代码按照数据帧字段配置内容逐一生成数据帧类的属性成员，并将每项属性成员在字节流中的位置记录在一个位置表中。数据帧类代码自动实现解包函数和打包函数，解包函数的输入参数为字节流，函数实现时按照位置表中的记录从字节流中取出数据并赋值给对应的字段属性；打包函数的输出为字节流，执行打包函数时，首先会创建一个内存缓冲区，然后按照位置表中的记录将字段属性的值逐一放入缓冲区的相应位置，然后将有效的缓冲区内容作为函数返回值输出。

在数据帧解包与打包报文的过程中需要处理几种特殊情况：

一是计算公式的处理：

在解包与打包过程中遇到计算公式时，会自动调用数据帧类的静态成员函数 `cacl` 进行数值计算，`cacl` 函数实现公式动态计算功能，`cacl` 函数的输入值为一个逆波兰式，返回值为公式计算结果数值。

二是动态解析的处理：

在解包与打包过程中需要进行动态解析时，会按照配置项中的内容对应生成一组 `switch case` 语句，用来动态选择字段解析方式。

三是数据验证的处理：

在解包过程中当字段定义有 `min`、`max` 属性时，会自动判断解包后的字段值是否落在规定的范围内，出现异常时会自动抛出异常。

数据帧类以静态成员函数的方式实现常用的 `check` 校验函数，当解包过程中的字段配置有 `check (crc)` 时会调用相应的校验算法进行验证，验证失败时会自动抛出异常；当打包过程中的字段配置有 `check (crc)` 时会自动调用相应的校验算法生成校验字段值。

### 3.5.3. 枚举定义代码

枚举配置项对应生成枚举定义代码，枚举定义具有公共访问权限，枚举定义代码在独立的代码文件中，在数据帧类和分系统类的实现中可以共用同一组枚举定义。

### 3.6. 异常处理

在协议配置组件运行过程中需要处置各种可能出现的错误,比如通信模块调用硬件驱动时可能会收到系统内核返回的错误码,数据帧解析过程中可能会出现解析失败或校验错误,生成的程序代码运行时也会有很多运行时错误;对于可能出现的错误协议配置组件以统一的方式进行处理。

当出现运行时错误时,协议配置组件首先会收集详细的错误信息,然后以抛出异常的方式向主程序报告错误,在抛出的异常对象中,包含完整的错误信息(错误类型、错误提示信息、报错位置、原始错误码等)。

## 4. 系统实现

### 4.1. 开发与运行环境

通信配置组件的开发环境基于 windows 操作系统 (Windows7 及以上版本),开发工具采用 Visual Studio 2017,其中协议配置模块采用 C# WPF 开发,语法检查与解析器为 C++开发的动态链接库,代码生成器是 C#开发的.Net 类库。其中.Net Framework 使用 4.5 版本,同时支持更高版本。

生成的程序代码支持 C#、C++, C#代码运行库为.NET Framework4.5 及以上版本,支持 Visual Studio 2017; C++代码支持 Visual C++和 Qt。

### 4.2. 关键技术路线

配置系统开发中,配置文件的代码格式通常会使用 XML、JASON 格式或自定义格式,本方案选用自定义格式实现配置代码,在语法检查和解析器的实现上相比其它技术路线要复杂,但却有其它技术方案所不具备的许多优势。

使用 XML、JASON 的主要优势在于:通用性强,大部分开发环境下都有可用的解析器,数据交互采用纯文本格式,适合系统间进行数据交互;其缺点也很明显:效率较低、人工输入较为复杂,不具备动态解析与计算功能。而采用自定义格式实现的配置代码,则可以针 XML 的不足,进行专门实现。通过分析用户业务

需求,为能生成更加通用的基础通信类库,以应对各种可能出现的复杂业务情况,本方案选用自定义格式配置代码进行实现,其中配置代码解析器采用凯云公司自主研发的成熟商业软件类库(ETest 软件的底层代码库)。

#### 4.3. 技术风险分析及应对措施

主要技术风险:

本项目主要风险在于交付周期很短,技术实现有一定复杂度。

应用措施:

- 1、本方案主要采用了凯云科技已有成熟产品中的核心模块进行二次开发;
- 2、成立专门项目组负责实施,由公司主管研发副总经理任项目经理,成员为公司在研发、测试等方向的核心技术骨干,可以确保项目按时按质交付。

### 5. 关于凯云科技

凯云联创(北京)科技有限公司(简称:凯云科技)核心业务是为军方、航空航天、中电、兵器、船舶、核工业、核物理、院校、交通、水利水电、通信以及金融等行业提供软件测试工具与测试服务,并承接软件开发、系统集成及信息技术服务。自主研发了超过 40 项拥有全部知识产权的软硬件产品,是国家级高新技术企业、北京市双软认定企业。

凯云科技自创建以来,始终致力于国产自主可控软件测试类产品的研究、开发及培训推广,为国内军用、工业、民用以及院校培训教学提供具有国际领先水平、通用性强、可靠性高、应用成本低的通用嵌入式系统测试及管理环境,同时积极引进国内外先进技术。主要产品覆盖软件测试业务全过程,其中有自主研发的嵌入式系统半实物仿真测试平台、嵌入式系统测试教学实训平台、软件测试项目管理系统、测试数据生成与管理软件等,引进产品包括各类源代码分析测试、性能及功能测试等工具。

基于长期在嵌入式系统测试领域的研究与创新,凯云科技目前已发展为国内嵌入式系统测试类产品的主要供应商,得到各行业用户的认可与支持。我们将一如既往坚持“以客户为中心,以创新产品和服务创造价值”,与用户携手,为国

内各行业装备研发水平与技术创新能力提高作出卓越贡献！

承制单位基本情况见附件十：承制单位基本情况表。

## 6. 同类项目执行情况

本方案中使用的配置代码解析器为凯云科技自主研发的代码解析器，是ETest软件的重要组成模块。

通用嵌入式系统测试平台（Embedded System Interface Test Studio, 简称：ETest）是针对嵌入式系统进行实时、闭环、非侵入式测试的自动化测试平台。该平台提供了嵌入式系统的半实物仿真测试环境，支持各类通讯接口，适用于嵌入式系统在设计、仿真、开发、调试、测试、集成验证和维护等各阶段配置项级别和系统级别的动态测试与验证。

ETest 提供了针对嵌入式系统的硬件在环半实物仿真测试环境，通过模拟待测系统的外部环境并产生信号输入到待测系统，同时获取并分析待测系统的输出信号，实现针对嵌入式系统的自动化功能黑盒测试。

ETest 具有适用范围广、通用性强、自动化测试程度高、扩展性好、携带方便、配置灵活、操作简单以及使用成本低等特点，能充分满足航空航天、武器装备、工业控制、汽车电子、仪器仪表等各领域嵌入式系统的测试需求。

凯云科技为行业客户提供了优质产品和服务，赢得客户的广泛赞誉，相关合同见附件十三：部分与项目相关产品及服务合同。

## 7. 安装、调试、验收方法及标准

本软件组件开发完成后，本公司将安排专职人员根据甲方项目要求做好安装、组件调试及与系统联合调试等工作。同时遵照甲方的验收方法和标准进行相关验收工作。

## 8. 质量保证

本软件组件开发将严格按照软件工程化标准进行，开发组织按以下步骤迭代施行：

- a. 需求分析
- b. 概要设计
- c. 详细设计
- d. 软件实现
- e. 单元测试
- f. 配置项测试

以上工作形成的阶段性交付物（包括需求规格说明、软件设计说明、软件测试说明、软件测试报告和软件用户手册）均需经过正式评审/会签。全部阶段工作完成后，将聘请具有 CNAS 资质的第三方测评机构进行评测，并提供合格有效的测评报告。

## 9. 培训、售后服务承诺

自本公司就交付的软件组件提供完整技术文档（纸质 2 份、电子版 1 份）并按用户要求提供多次现场培训，确保用户正常使用全部功能。

自项目验收之日起，提供 3 年技术支持服务，服务内容包括电话咨询、远程协助、上门技术支持等，提供全年 7\*24 小时远程支持服务，严重问题可按用户要求 2 小时内到达现场。

## 10. 交付物

本项目最终交付物包括：

- 通信接口及协议配置组件源代码及可执行程序
- 用户手册电子版与纸质版
- 需求规格说明电子版与纸质版



- 软件设计说明电子版与纸质版
- 软件测试说明电子版与纸质版
- 软件测试报告电子版与纸质版
- 软件第三方测评报告电子版与纸质版

## 11. 交货期

自合同签订一个月内，向用户交付基于 C#语言的通信接口及协议配置组件，以及配套的软件用户手册。

合同签订起两个月内，向用户交付全套完整的通信接口及协议配置组件，以及全套文档，含全部交付物。

## 四、附件