需求概述

1. 以下所有需求针对C# 以及C++（Visual C++ 和 Qt），即必须支持C# 和 C++ 两个平台，以及必须支持Windows和Linux两个操作系统（主要是CAN在两个操作系统下的调用方式可能不同）。其中 C# 基于 .NET Framework 4.5 或以上版本开发；
2. 编写 TCP（包括服务器和客户端）、UDP、串口和CAN的底层通信类；
3. 编写协议配置软件，可对协议进行新建、修改和删除；
4. 根据配置的协议，编写 IDE 插件或者独立软件，自动生成相关代码；
5. 所有模块和软件均需提供源代码。

底层通信模块

编写 TCP Server、TCP Client、UDP、串口、和 CAN 的独立的底层通信模块。

发送和接收均基于二进制方式（即不用提供文本方式）。

TCP Server、TCP Client和UDP在事件参数中还需提供发送方的 IP 地址和端口，CAN需要提供发送方的节点ID。（C++ 平台要求类似，除了事件方式有区别。）

CAN通信模块至少需要支持广州致远和研华的 CAN 设备（同一个厂家不同的型号的CAN设备的API基本相同），我们目前已使用过的CAN设备包括：

广州致远USBCAN-2E-U（USB转CAN）

广州致远PCIe-9221（PCIe CAN卡）

研华MIC-3680（CPCI CAN 卡）

协议配置模块

针对我们目前使用的协议框架，编写一个协议配置模块，可对不同分系统间的协议进行配置。

协议存储文件格式：XML 或 JSON 或其他自定义文本格式。

每个项目（课题）的所有协议存储为一个文件，不同项目之间的协议没有直接关系。每个项目设置一个英文ID和中文描述，英文ID需符合一般的代码的变量名命名要求（即只能是英文、数字和下划线，不能以数字开始），英文ID在之后生成代码时用作命名空间。

每个项目下，有两个或以上的分系统，每个分系统设置一个英文ID和中文描述，英文ID同样要符合代码的变量名命名要求。每个分系统分配一个标识码，标识码类型为 byte，即范围为 0x00~0xFF。

选择两个分系统后，首先需要选择两者间使用的通信方式，如是TCP、UDP、串口还是CAN。其中TCP必须指定哪一方是服务器，哪一方是客户端。CAN可以指定是哪一个CAN设备（即目前是广州致远和研华厂家的哪一个型号的设备），也可以指定为“未知”，因为有可能某个分系统运行的是单片机程序。

另外串口需要指令双方通信的波特率、数据位、停止位和校验位。

CAN通信方式需要指定双方的节点ID，由于CAN通常用来在总线接入多个相同的设备（节点ID不同），因此CAN方式其中一方的分系统可支持多个设备，每个设备需指定自己的节点ID。

TCP和UDP的IP地址以及端口设置、串口的串口名设置因为与协议无关，所以不在此设置（之后在自动生成代码时以构造函数参数传入）。

确定通信方式后，可为其配置协议。例如选择了分系统A和B后，需要配置A发往B以及B发往A的指令。

TCP、UDP和串口的通信方式基于我们的协议框架，CAN通信方式由于每包只能发送8个字节，字节数有限，所以未做框架，直接配置里面的数据内容。

协议框架说明：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 含义 | 数据类型 | 字节长度 | 备注 |
| 帧头 | 数据源标识码 | byte | 1 | 填入分系统ID |
| 接收方标识码 | byte | 1 | 填入分系统ID |
| 数据内容标示 | byte | 1 |  |
| 包长 | byte | 1 | 包长是所有字节的长度，即包括帧头、数据内容和校验的长度 |
|  | 数据内容 |  | 可变 |  |
|  | 校验字 | ushort | 2 | 除校验字外所有字节之和的低16位 |

协议中所有字段都是低字节在前（Little-endian）。

注意：“是否是指令反馈”和“是否定时发送以及发送频率”和“发送时数据内容是否产生字段”三个选项为互斥关系，最多只能三选一，也可以不选中任何一项。

数据内容配置即配置此包发送的数据，可不含任何数据，或者包含多个数据。

每条数据必须要配置的项目包括：

|  |  |
| --- | --- |
| 名称 | 说明 |
| 英文ID | 必须符合代码变量名命名规则 |
| 中文描述 |  |
| 数据类型 | 见下面说明 |
| 浮点数发送系数 | （仅针对整数类型）某些协议对浮点数发送时是乘以一个系数，取整后发送 |
| 浮点数原始类型 | （仅针对整数类型）原始类型是float还是double |
| 最大有效值 | （bool、enum和预留类型 无此设置）默认为空 |
| 最小有效值 | （bool、enum和预留类型 无此设置）默认为空 |
| 默认值 | （预留类型 无此设置）默认为空，但一旦设置了“最大有效值”或者“最小有效值”后，必须设置此数值 |
| 单位 | 默认为空 |

注意：最大有效值、最小有效值和默认值均针对发送时的数据类型（即如果是浮点数乘以系数后发送，以发送时的整型为准）。

可配置的数据类型包括：

|  |  |  |  |
| --- | --- | --- | --- |
| 数据类型  （按照.NET 的类型） | 占用字节 | 需要特别配置的选项 | 说明 |
| bool | 占用位数不固定 | 占用的位数  True对应数值  False对应数值 | 占用的位数的可选范围为 1、2、4、8  True 和False对应数值不能超过占用的位数范围  （有可能 11表示True，0表示False） |
| byte | 1 |  |  |
| sbyte | 1 |  |  |
| ushort | 2 |  |  |
| short | 2 |  |  |
| uint | 4 |  |  |
| int | 4 |  |  |
| ulong | 8 |  |  |
| long | 8 |  |  |
| float | 4 |  |  |
| double | 8 |  |  |
| enum | 占用位数不固定 | 配置枚举包含哪些值（每个值需要配置英文ID和中文描述） | 枚举的类型为byte，即范围为 0~255 |
| 预留 | 占用字节数不固定 |  | 预留1个或多个字节 |

因为分系统有可能同时控制多个数量的子系统，发送状态时会将多个子系统的状态循环，因此协议配置需要支持数组方式，只需要配置一个子系统的状态，然后自动循环多次。

很多时候，子系统的状态是由多条包返回，因此允许在分系统内定义一个或多个子系统，子系统需要配置的项目包括：

英文ID、中文描述、含有哪些数据（每条数据又包括英文ID、中文描述、数据类型、最大有效值、最小有效值、默认值和单位）

在配置完子系统后，在配置协议时，可从子系统中直接添加引用。

注意：TCP、UDP和串口的通信方式，数据内容字节长度总和不能超过 249，CAN通信方式数据内容长度总和不能超过8。

自动生成代码

编写一个自动生成代码软件（可与协议配置软件合并）或者 IDE 插件（Visual Studio 必须支持 2017 版本），在选定一个项目后，选择其中的一个分系统后，自动生成相关代码。每一个对方分系统生成一个类，代码根据协议自动完成包的组包发送和接收解包，如果有子系统，自动生成子系统类。

所有的接收包，都需要根据包内的数据内容生成相应的变量；发送包，则根据包的“发送时数据内容是否产生字段”来确定是否生成相应的变量。

每个发送包均要提供一个单独的方法（函数），供外部调用，格式为 SendXXX(…)，XXX 为该包的英文ID。

每个接收包收到之后，需要通知外部已收到该包，C# 中以事件方式通知，事件参数为枚举，枚举包含每个接收包的英文ID。

分系统的通信设置（比如TCP和UDP的IP和端口设置、串口的串口名）以构造函数的参数传入。

所有的分系统的父类为 IaeSubSystem，所有的协议配置代码都在 IaeSubSystem 类以及分系统子类中完成。IaeSubSystem 的父类为 SubSystem, 该类中只包含基本的属性，包括 Name 和 IsConnected 等。（IsConnected 根据是否收到数据包为准，通常情况下，协议中都有定时发送的包）。

在 C# 中，所有生成的类均使用 partial 关键字。

SubSystem 和 IaeSubSystem 类位于 “项目名.SubSystems” 命名空间下。

分系统类的命名空间为：”项目名.SubSystems.分系统名（复数）”。比如项目名为 Abc，分系统英文ID为PowerSupply，则该分系统的命名空间为：Abc.SubSystems.PowerSupplies。

在 C# 中，所有的类的命名空间与文件夹结构要匹配，但是在 SubSystems 文件夹下建立一个 Protocol 文件夹，所有的类文件均放在 Protocol 文件夹下。例如 Abc.SubSystems.PowerSupplies 命名空间下的 PowerSupply.cs 文件位于 “Abc -> SubSystems -> Procotol -> PowerSupplies“ 文件夹下。

在C# 中，所有的变量不是直接使用相应的数据类型，而是将其封装为类，类名为 Parameter<T>，数值通过 T Value 属性访问。所有的接收包的变量，传入的 T 使用相应的 Nullable 类型，比如 bool 数据，实际实例化的类型为 Parameter<bool?>，所有的变量默认值为 null，收到数值后更改数值，在分系统掉线后，变量数值又自动变回 null。所有发送包，如果有定义变量，则直接使用相应的数据类型（即不使用 Nullable 类型）。

Parameter<T> 类中，包含一些基本的属性，比如Name，自动填入中文描述。还需提供 ValueChanged 时间，在数值变化时触发事件。

生成代码的类模板允许修改。