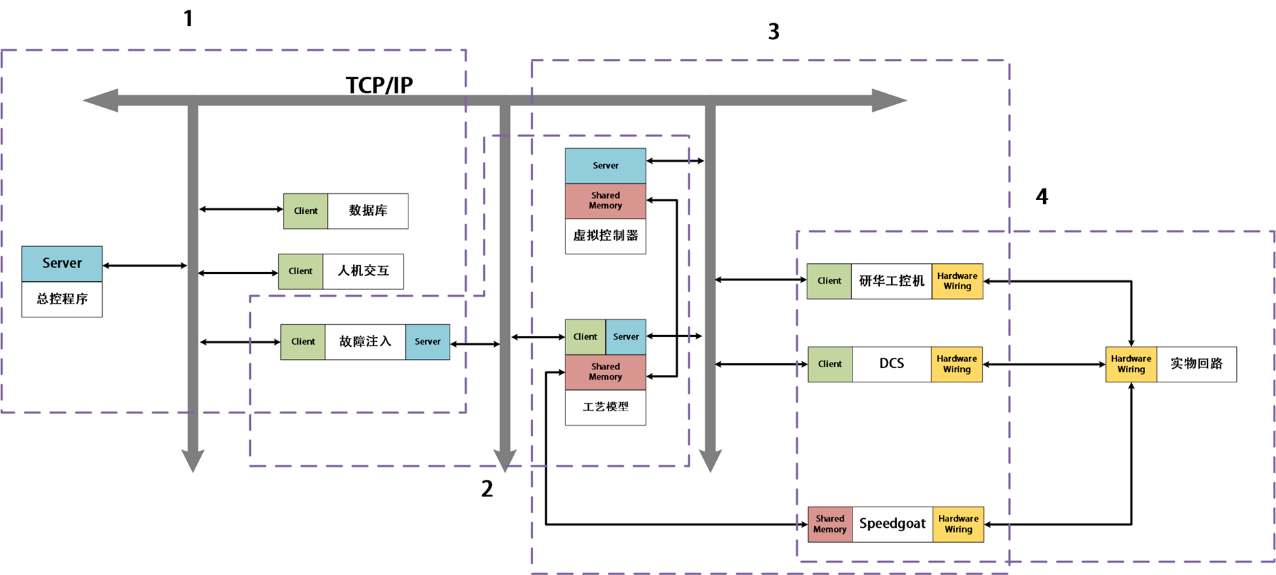
**工程说明文档**

**版本历史**

|  |  |  |
| --- | --- | --- |
| 版本 | 撰写人 | 备注 |
| V1.0.1 | 王文韬 | 初始版本 |
|  |  |  |

1. 需求描述

实验室设计框架如图：



实验室需求分解：(待补充)

根据实现实验室设计框架，目前已经实现以下功能：

1. 控制中台
2. crc校验算法
3. 共享内存
4. 工程目录

虽然在实验室系统中是多个应用程序彼此独立，但是将所有程序统一存放在一个文件夹中十分重要。在同一个文件夹中统一管理，可以更方便版本更新，可防止部分程序更新，部分程序未更新导致的错误问题，保证在迭代过程中，始终有可正常运行的版本。同样统一管理也更利于进度管理，方便项目进度的安排。统一管理也方便成员之间相互review 代码，更容易定位错误原因，互相发现问题，互相学习进步。

**希望大家尽快将自己对应代码按照工程目录说明，添加至对应位置**，我同样会将更新后代码上传到github仓库中，方便版本管理。

Github仓库地址：<https://github.com/valourshadow/xjtu_810_nrcl_lab_f1>。以下为目录说明：

1. build目录

build目录存放编译相关中间产物，包括但不限于：

1. cmake文件产生的中间文件等。
2. cmake编译选项文件选项。
3. IDE生成对应非可执行程序的中间文件，类似于.prj文件等。（在完成代码的情况下，建议使用非IDE编译，使用IDE编译在代码移植过程中会不可避免的出现环境配置问题）。
4. 其他编译中间产物，类似于.o等文件。

build下目录因为是编译产物，所以不会进行版本追踪，所以不要将任何非自动生成的内容至于build目录之下。

1. inc目录

inc目录放置工程头文件，inc目录会被版本追踪。

项目组成员需要根据不同功能所需，在inc建立不同子文件夹，存放对应头文件或者配置文件。**要求除nrcl\_config头文件之外，不同子文件夹中的头文件不可相互依赖**。不同子文件夹中功能需要相互解耦，方便后续针对单个模块进行升级迭代。

nrcl\_config.h是配置头文件，多个程序所需宏定义，常量信息，数据结构别名需要在该文件中定义，禁止程序中出现魔法数字，造成代码理解困难。

inc目录下现有：nrcl\_config.h 与crc文件夹。nrcl\_config.h文件中定义了u8等数据变量。crc文件夹中包含了crc.h文件，是crc算法所需头文件。假设，张三需要实现读取数据的功能，这个功能需要一些头文件。那么他需要在inc文件夹中，新建readdata的子文件夹（防止歧义，文件夹名需要描述功能），并将对应头文件放入readdata子文件夹中。

1. doc文件夹

doc文件夹有两个子文件夹，用来放置对应文档，需要在程序更新时，同时更新文档，防止代码与文档不匹配所造成的困扰。doc文件夹中还有工程总体的说明文档，即本文档，每次更新需要在本文档中加入对新功能的大概描述。

design\_report文件夹下放置对应模块的设计文档，设计文档命名为XX设计报告，需要依次包含以下几个部分：

* 报告题目、版本历史
* 需求或bug描述
* 该功能所需头文件、源文件以库目录等
* 程序实现方式以及程序框图
* 所需数据结构（可选）
* 程序自测试方案。

test\_report文件夹下需要防止对应模块的测试报告以及系统测试报告。模块测试报告命名为XX测试报告，测试项目应大于或等于设计报告中的内容，该测试报告需要包含测试时间，测试方法，测试期望结果，以及测试结果几个部分。

系统测试报告为系统完成后发布测试，具体形式等后续指定。

文档比代码更重要，写文档的过程就是梳理思路的过程，也是对自己工作复盘的过程。没有一个详细的设计方案，在写代码的时候总会觉得没有思路，逻辑很混乱，不知道具体要怎么解决。**因此，要求文档必须有并且需要提交**。

1. src文件夹

src文件夹放置对应程序或者库的源文件，不同功能在其下建立对应子文件夹即可，要求程序间的功能需要解耦，可复用的部分需要在单独的子文件夹中。命名方式与inc文件夹相同。可参考inc文件夹建立子文件夹过程。

1. lib文件夹

lib文件夹放置第三方库。例如libmodbus，qt等。同样的，需要根据不同的库，在lib文件夹下建立对应子文件夹，将库放置进去。库尽量选择成熟的官方库，可以避免很多意料之外的错误。

1. out文件夹

out文件夹有bin和lib两个子文件夹，用来存放对应的可执行文件和库文件，out文件为编译生成，不会对其进行版本管理与追踪。out文件夹为最终发布或者部署的程序文件。

bin文件夹用来放置生成的可执行程序，比如exe格式，或者对应脚本。在编译时，需要指定可执行文件的生成路径为bin，bin文件夹下不用建立子文件夹。

lib文件夹用来放置对应库文件，有两个子文件夹。nrcl\_part文件夹用来放置由成员代码生成的库，比如通讯代码库，数据协议解析库等。这个部分同样由编译产生，无需建立子文件夹。third\_part文件夹用来放置第三方库，这个库为第五小节中lib文件夹的拷贝，仅在程序发布或者部署时需要。因此可不关心该文件夹，在发布时由发布人统一处理拷贝即可。

根据以上描述，可以得出一个功能的实现流程应为：

1. **撰写对应设计报告**
2. **在组群中发布报告，并邀请成员review报告**
3. **报告通过则编写代码并提交报告，不通过则转至第一步。**
4. **更新程序至最新版本，将自己的代码提交到对应文件夹下。**
5. **提交测试报告并更新工程说文档。**
6. **在github上提交代码，并在组群通知成员review代码。**
7. 已实现功能
8. 控制中台

控制中台实现的功能为接受数据库，人机界面以及故障注入系统的数据请求，以及对数据库、人机界面以及故障注入系统的数据请求进行响应。接收或者发送数据，对数据进行合法性校验，并且处理或者分发对应数据。

控制中台的实现使用了网络编程技术以及多线程异步编程技术。控制中台在c/s模型中为server端，是被动等待其余模块请求，若其余模块没有请求，应该保持静默。这一版本的实现是基于TCP/IP的套接字实现，没有加入其余用户层协议。控制中台在死循环中阻塞监视套接字请求，如果有请求，则启动一个线程去判断数据的合法性并解析数据，根据数据的解析结果，跳转至对应的处理线程。现在的处理线程主要有三个，ui处理线程，数据库处理线程已经故障注入处理线程。

控制中台目前已经可以与client端通信，并且跳转至对应处理线程，但是具体处理需要依赖于数据库，ui界面，故障注入等功能的接口函数。

1. crc校验

crc校验主要是为了校验网络通信所传输数据的准确性，防止因为传输数据的错误引起的系统运行问题。crc校验算法根据数据包计算出一个特殊的值。发送方先在本地计算出这个值，将他加入到传输数据中；接收方接收数据之后，通过同样的算法对接收的数据包计算crc值，接收方通过比对计算的crc值与接收的crc值是否相等就可以判断出数据传输是否发生错误。

crc的实现主要是根据crc算法的原理进行计算，编写函数。

目前已经实现不同种类的crc8校验和crc16校验。

1. 共享内存

共享内存为进程之间通讯的技术，其原理就是将两块个进程的逻辑地址相同的虚拟内存映射到同一块物理内存上去。共享内存是进程间高效的通讯方式，适合大量数据传输的场景。例如不同工业软件之间的通讯。

共享内存的实现是依赖于windows的自带库，本版本的共享内存函数仅可在windows系统上运行。

目前已经实现了共享内存读函数和共享内存写函数。在使用时需要根据工业软件的不同，去调用函数。