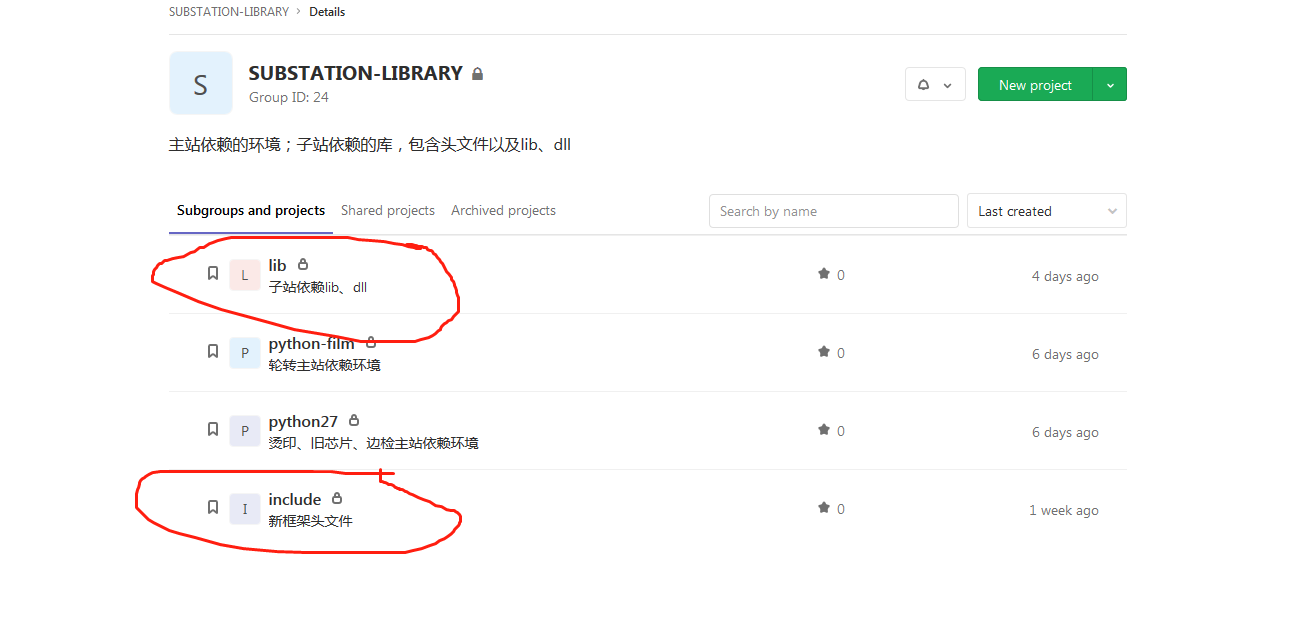
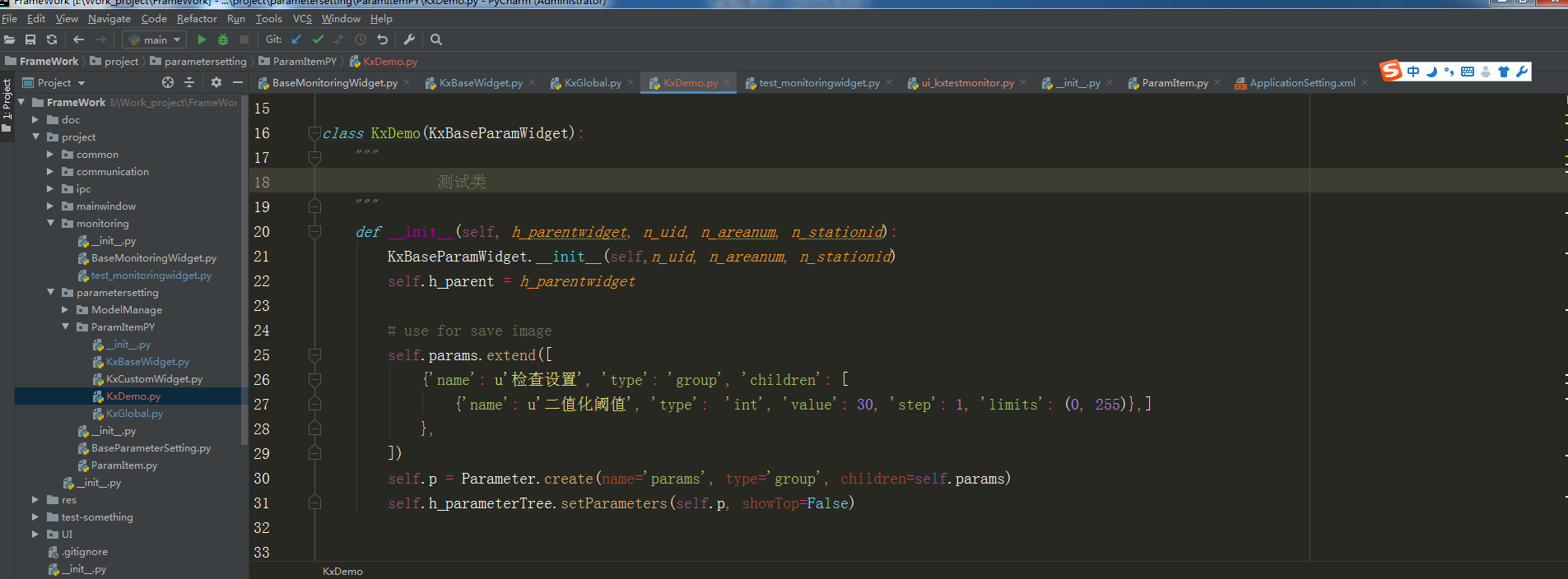
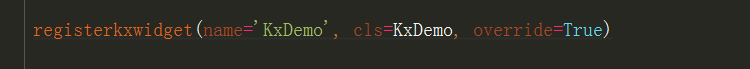
# 主站部分

1. 将Framework从服务器上拉取，然后删除git仓库，新建git仓库，设置文件名，使用readme文档中方法拉取项目所需子库。或者在Frameword目录下，拉取kxpyqtgraph、library等库，因为framework依赖于这些库（开发时不要直接在这些库上改）；目的是为了保证项目是项目，底层库是底层库，在保证项目的进行时，如果kxpyqtgraph、library有更新，可以直接更新而不改动项目。库拉完后删除.gitmodules文件，然后git add .，也即把当前库一同提交到这个新建工程。

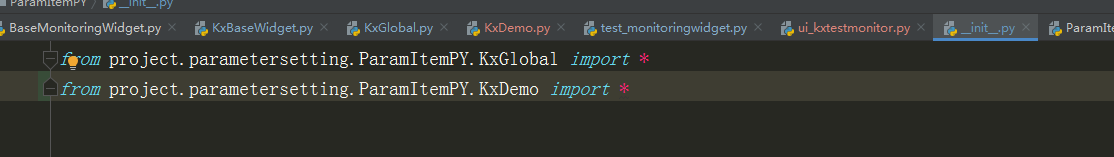
0.1 对于子站，库取服务器中include 、lib两个项目，拉下来后解包，这些都是公共库。算法库在另外的组。



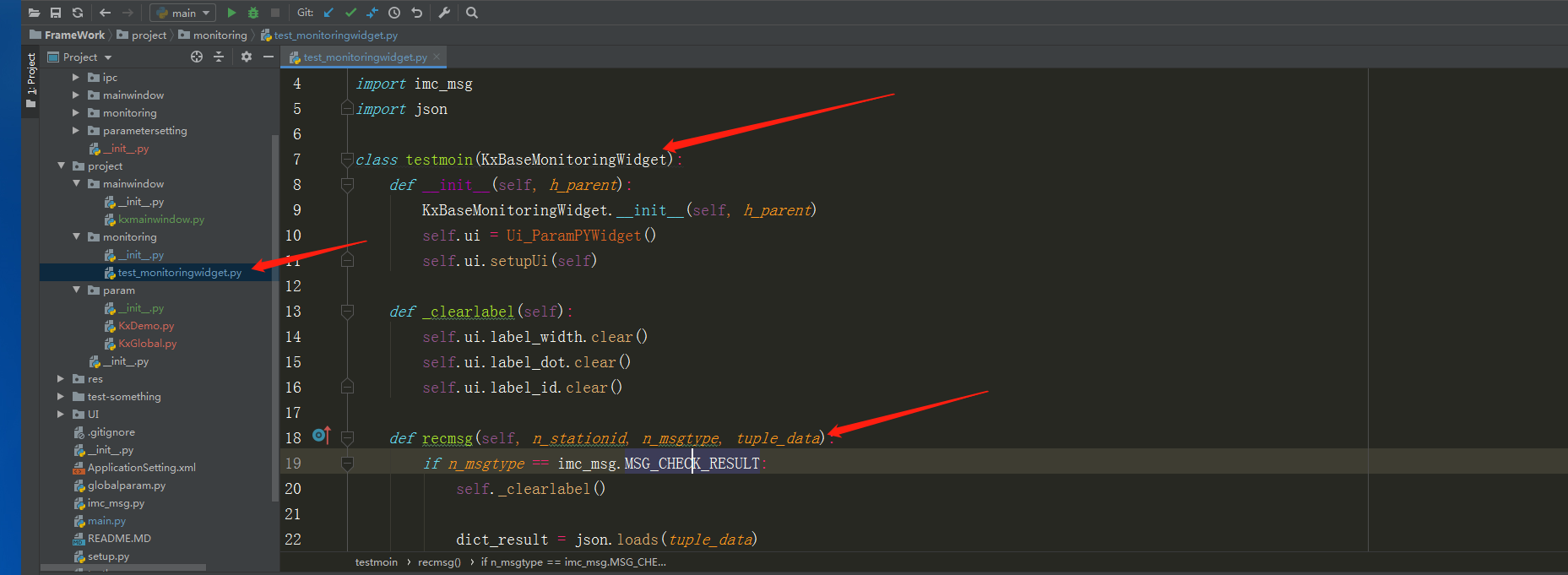
1. 继承KxBaseParamWidget写参数文件,放于project/param文件夹下
2. 注册类名registerkxwidget



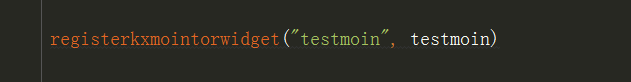
1. 在project/param 的\_\_init\_\_文件中import进新写的参数文件



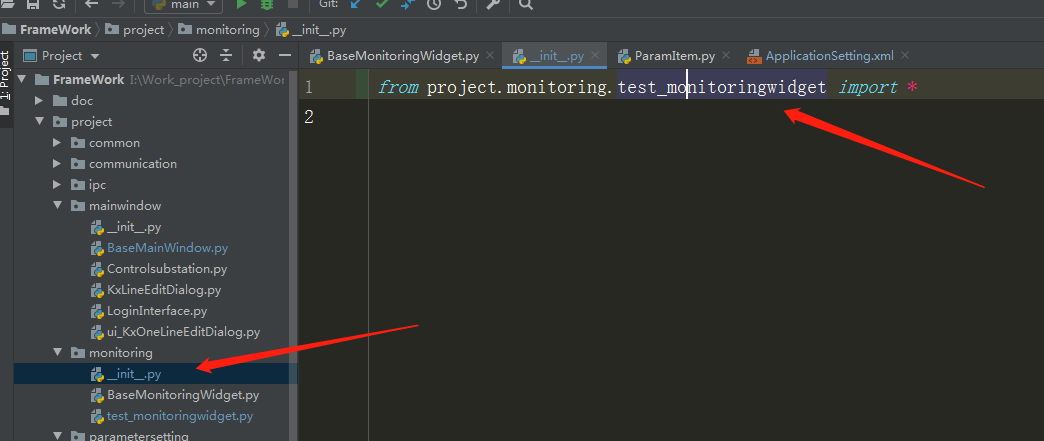
1. 画实时显示界面，qtdesigner、pyuic，
2. 继承于KxBaseMonitoringWidget,重写recmsg，实现功能,把文件放在project/monitoring



1. 注册类名registerkxmonitorwidget

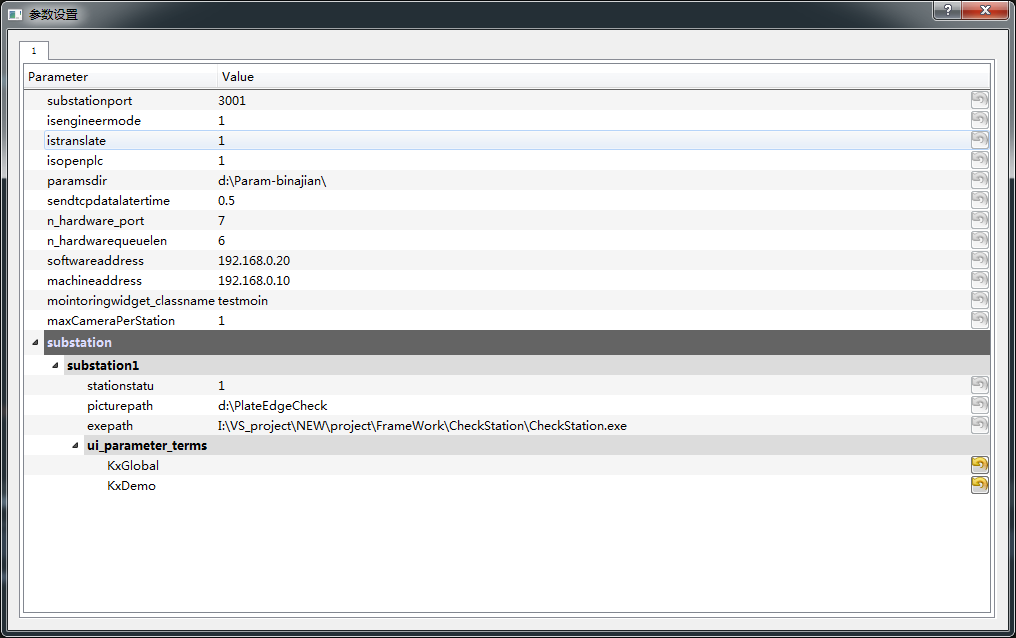


1. 在monitoring的\_\_init\_\_文件中import进新写的界面文件



1. 启动程序，打开主站配置，然后点下任意一个参数（选中界面），按下Alt键

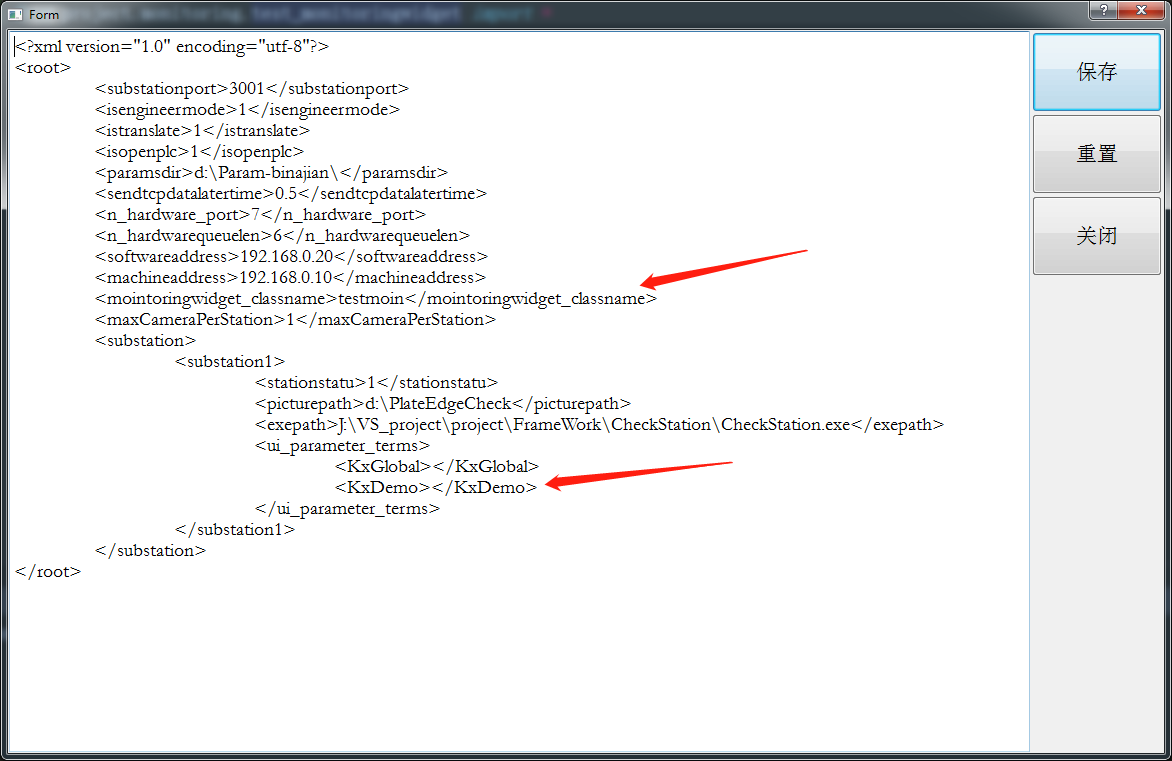




1. 如下，进入编辑界面。根据自己的需求进行修改，如果是参数文件，则更改<ui\_param\_terms>下的内容，比如你自定义了一个参数类叫”hongparam”， 那么就把<KxDemo></KxDemo>改为<hongparam></hongparam>，当然，如果你是打算在原先的基础上继续加参数，则直接在KxDemo后追加;

如果是实时显示界面需要修改，比如你自定了一个”hongmoni”，则将<mointoringwidget\_classname>testmoin</mointoringwidget\_classname>改为

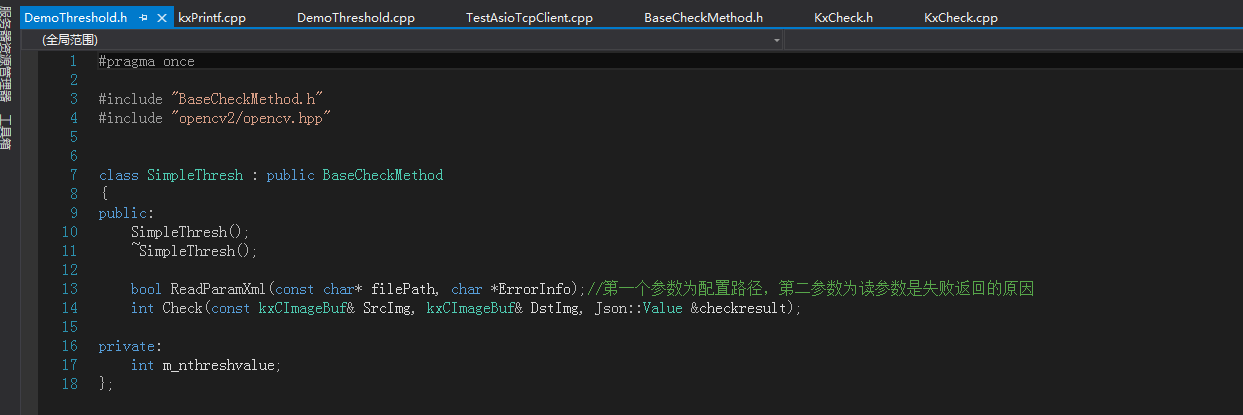
<mointoringwidget\_classname>hongmoni</mointoringwidget\_classname>



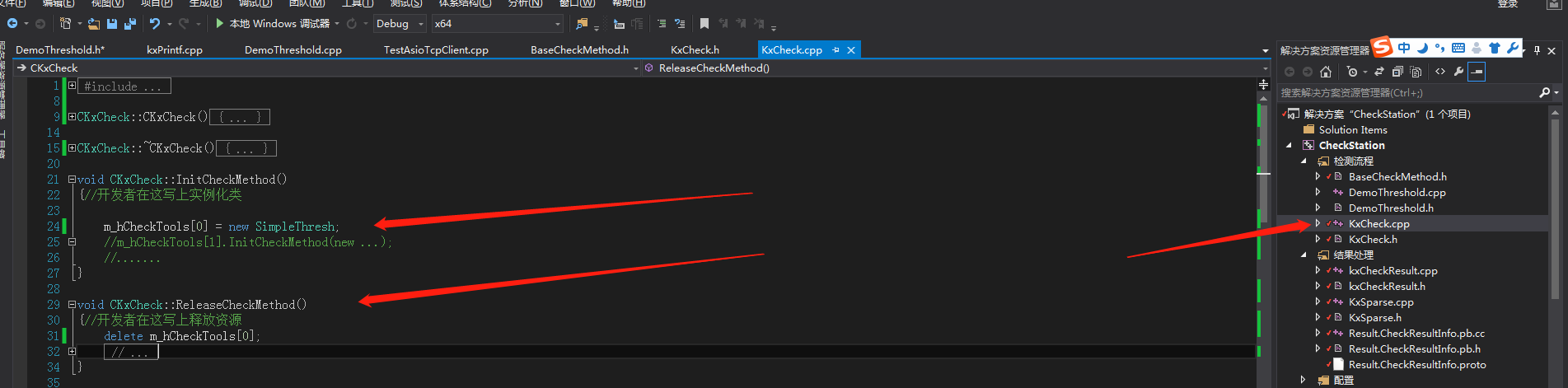
关于以上的demo，看KxDemo.py、test\_monitoringWidget.py例程

# 子站部分

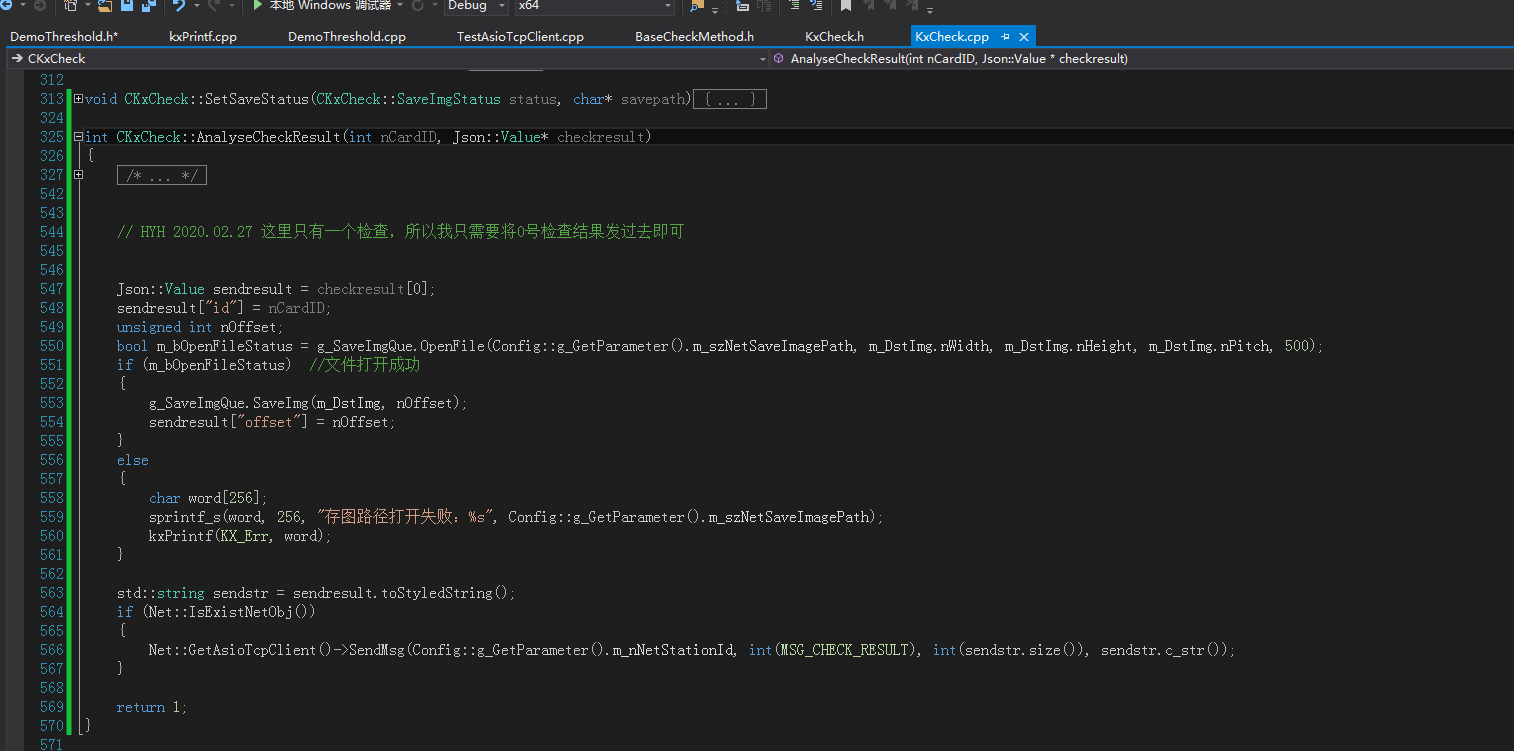
1. 编写自己的检查算法类，继承于BaseCheckMethod，必须实现抽象父类的方法，用以读取参数、检测图像。



1. 在KxCheck.cpp中加上写的算法类的实例化



1. 在KxCheck.cpp的AnalyseCheckResult中写上对处理结果的判断，并将处理结果发送到主站中去

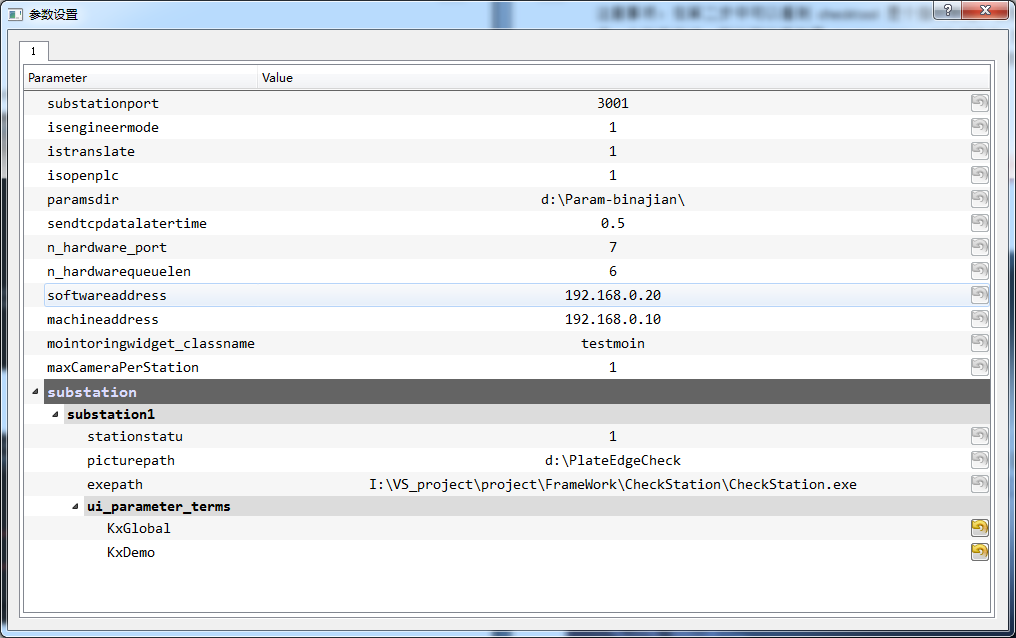


注意事项：在第二步中可以看到checktool是个指针数组，数组中的指针指向的是检测算法的基类，现在只考虑一个检查方法，所以可以看到是checktool[0] = “检测类”；当主站有多个检查方法的时候，也即主站中一个站点具有几个参数界面的时候，就需要checktool[1] = “检测类”..........。

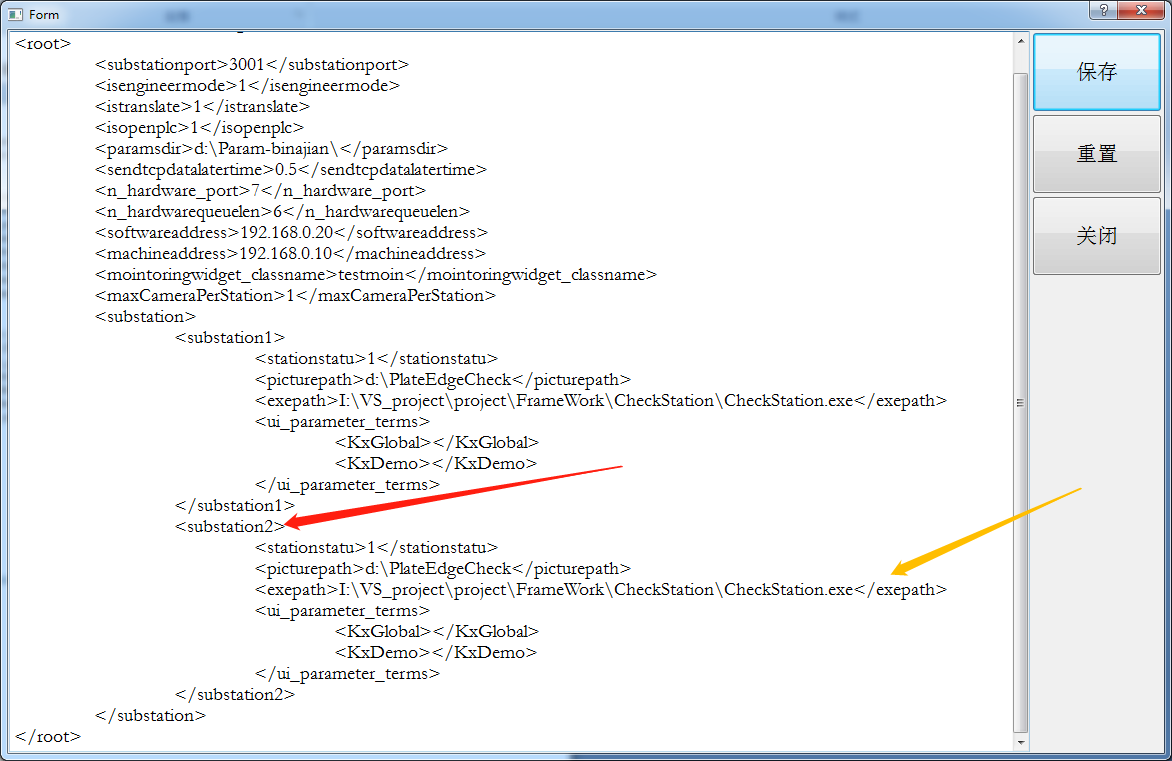
# 上面的设计都是为一对一而设计，下面考虑一主多子

1.首先进行配置，打开主站程序，选择左上角的主站配置



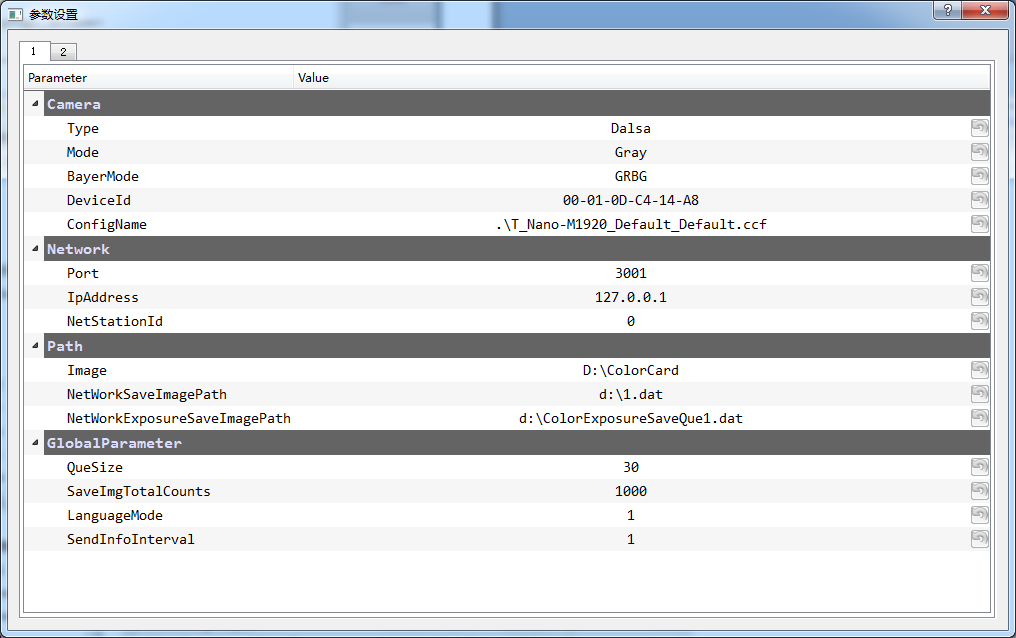


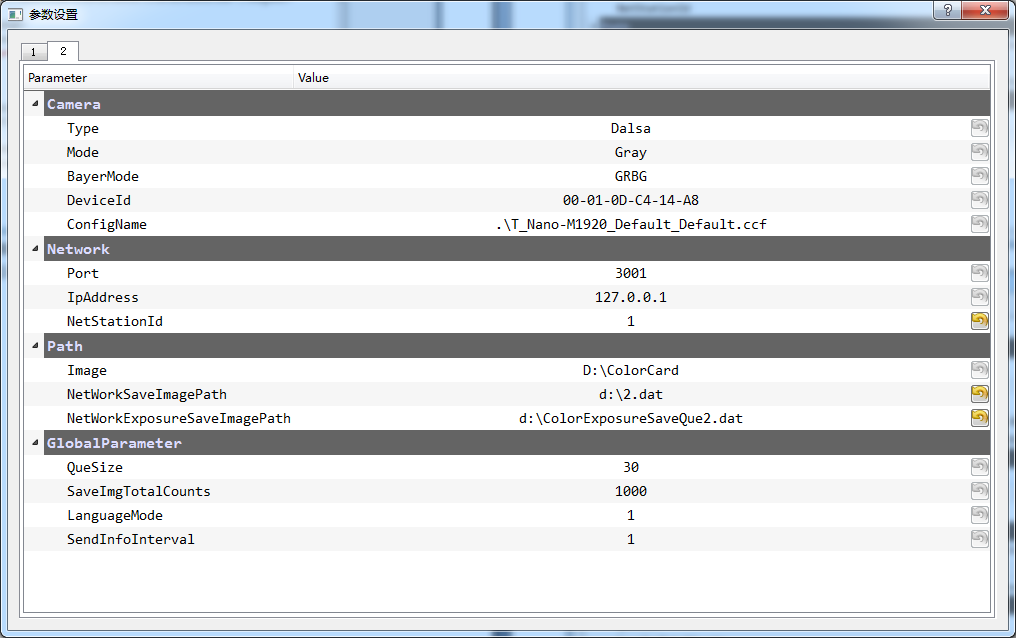
打开之后选中参数设置界面，按下Alt键



出现主站编辑窗口，如上图所示，红箭头表示添加的站点内容，黄色箭头指的是新站点的启动程序位置，以上非常重要，比如注意格式，正确添加。点击保存，退出设置即可。

2.打开子站设置





这里每个参数的意义在代码中有，这不赘述。主要关注NetStationID，这个站点号必须从0开始，依次叠加1，如上两图，两个站点的设置分别为0,1。

另外两个路径，NetWorkSaveImagePath是检测过程中子站给主站传的路径，这里也必须唯一

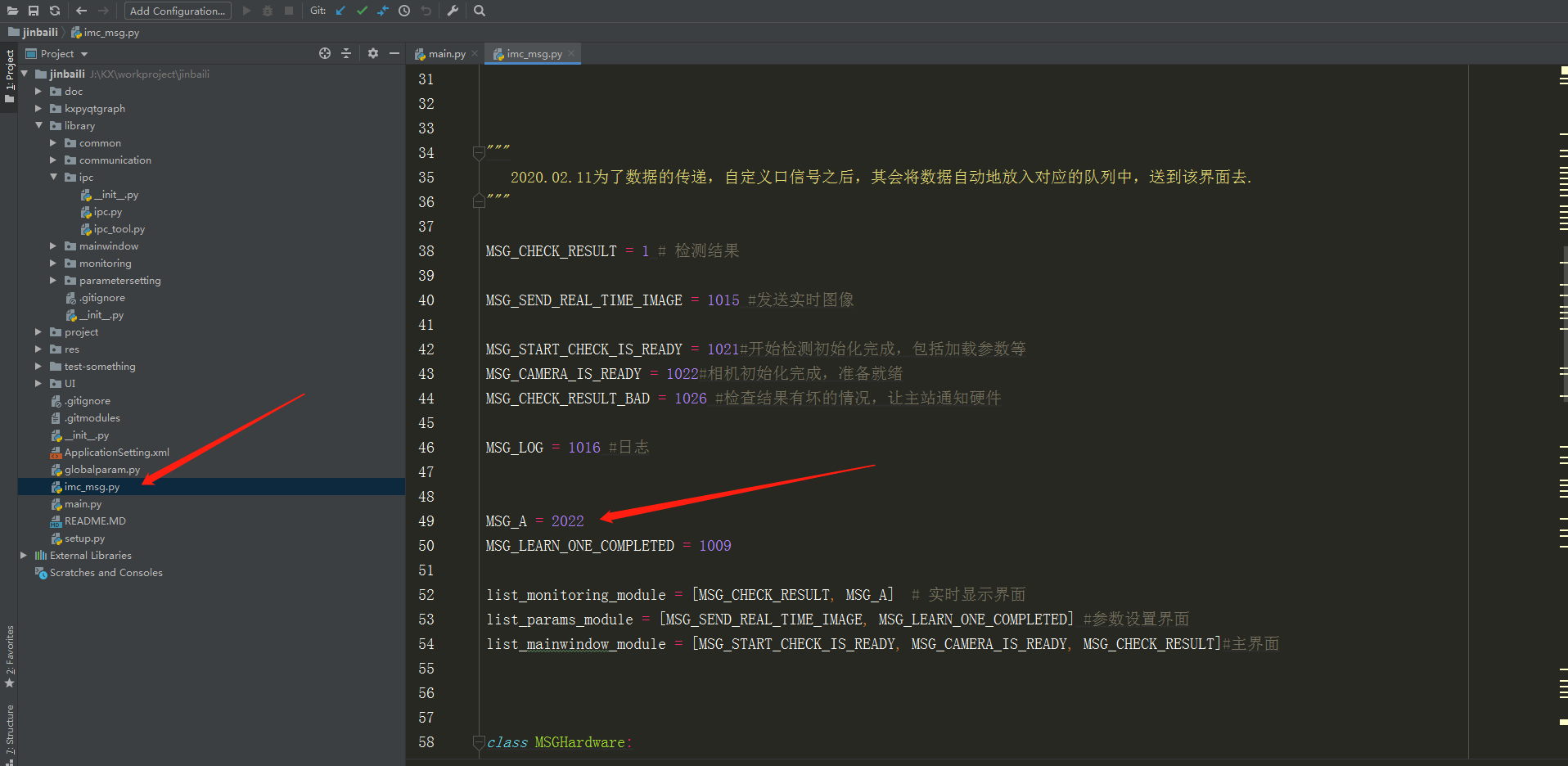
NetWorExposureSaveImagePath是对相机采集实时图像的路径，也必须唯一。

设置完毕之后保存设置，打开主站。

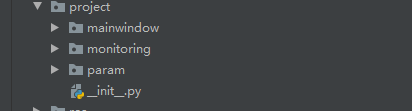
# 定义传输消息方法

子站定义这里不说，主要为主站，主站的定义较为复杂，但因为框架经过优化，只需要按步走即可快速完成子主通信。

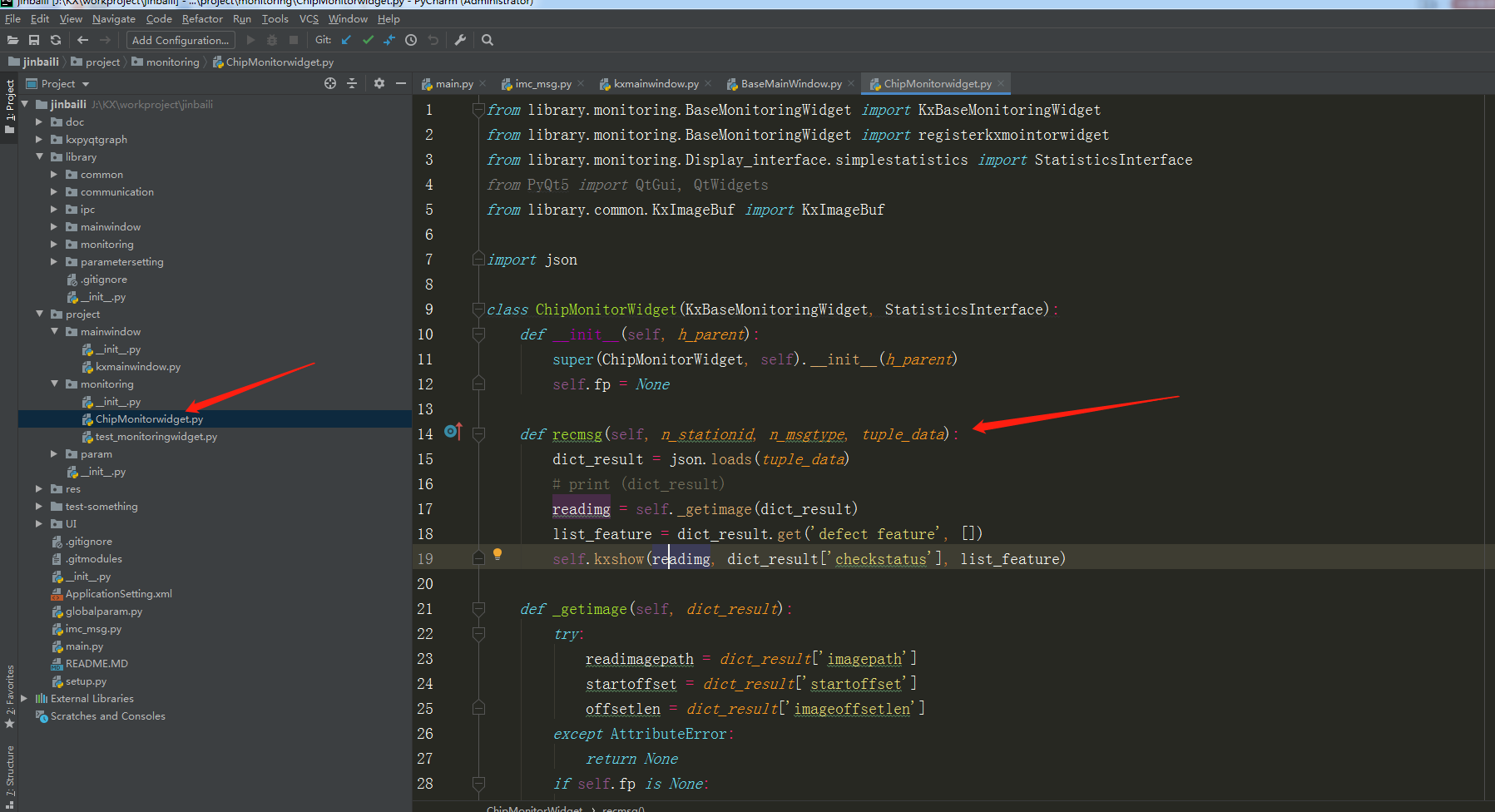
1. 找到imc\_msg.py，增加对应的消息头



1. 将消息头放入 list\_monitoring\_module/list\_params\_module/list\_mainwindow\_module三个其中一个，也可都放，其实原理很简单，这三个列表决定了消息会发往哪个界面，三个界面分别对应实时界面，参数界面，主界面，也即对应工程中的这三个



1. 找到对应的py，重写recmsg函数，这里以实时界面为例，如下图



# Git submodule功能

## #首次添加submodule

git submodule add 仓库地址

git commit 提交

## #拉取方式

直接git clone 拉取之后并不会自动下载submodule

使用git clone 主仓库 --recursive

忘记--recursive 需要git submodule init && git submodule update

## #更新submodule

git submodule update --remote