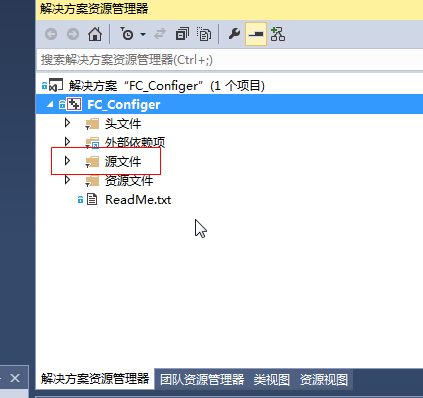
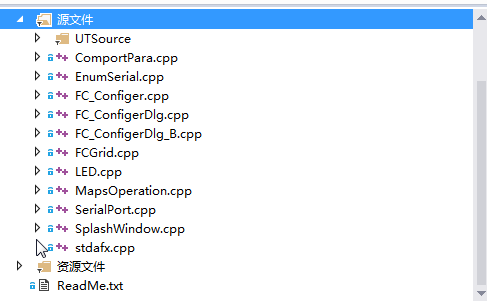
# 1,源文件说明:





## 主要文件：

我们一般修改的逻辑都在FC\_\_ConfigerXXX.cpp（ FC\_ConfigerDlg.cpp, FC\_Configer\_B.cpp）与MapsOperation.cpp这个几个文件中。这个几个文件的内容都是类FC\_ConfigerDlg的各个方法的实现（注：可以看出这是不合理的，但就这样吧）。

## 界面控件：

UTSource是UtilitySource的简写，这里主要实现的是界面控件。

FCGrid也是为界面服务重写的

界面对话框：

ComportPara：设置串口波特率界面

SplashWindow：启动界面

FC\_Configer.cpp：MFC入口，类似main函数

Stdafx：预编译的cpp，LED：不知道做啥的。

## 串口相关：

SerialPort：调用windowAPI实现读写串口

EnumSerial：串口需要的一些方法。

# 2，功能说明

注：此处可能说不清楚，若需要更明白可以问王佳星。

## 软件界面图：



其中有读取和写入两种操作

## 读取：

以圆圈标识，把数据从硬件（比如fpga）通过串口读到界面上。

点击一个读取按钮的后，读取过程分为4步：1，生成读取指令（指令在代码中是类PC\_CMD\_ITEM）放到指令列表（cmd\_item\_list）中，并且通过串口发送其中一个指令到硬件（把cmd\_item\_list中的一个指令发送出去）。2，硬件收到数据，按照协议（协议详见“上位机配置软件相关要求xxx.doc”）解析之，然后把请求的数据通过串口返回。PC有个定时器（OnTimer(UINT nTimerID)）不断的从串口读取数据，读到一个完整的数据，就按照协议（协议详见“上位机配置软件相关要求xxx.doc”）把数据写入到对应的buffer中（此处buffer在代码中主要是指Card\_Flash\_Table，m\_nPriorityEth，m\_nCAN\_Send\_DID，m\_nSwitchID等等。写入的过程方法为ProcessRcv(UINT rcvdata)）。3，通过串口发送下一个指令到硬件，重复1、2步骤直到指令列表发送完毕。4，界面显示的数据就是从这个buffer中获取的。

## 写入：

以矩形标识，把界面的数据通过串口写入到硬件（比如fpga）。

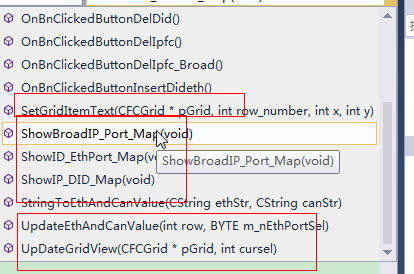
把界面的数据写入到对应的buffer中，发送buffer中的一条数据的方式是按照协议通过串口发送给硬件。接收到ACK后，再发送第二条信息。

## 方法讲解：

## 显示数据到界面：

ShowIP\_DID\_Map();是把buffer的IP\_DID数据显示到IP\_DID界面上

同样的还是有



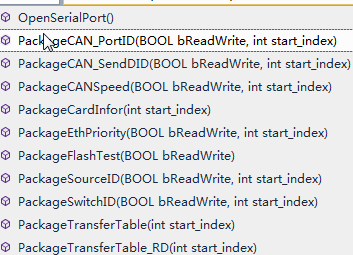
都是显示数据到界面的方法。

## 包装读取或写入数据的方法：

PackageTransferTable\_RD是把界面上4个表格的数据一键从硬件读取上来，

PackageTransferTable是把界面上4个表格的数据一键写入到硬件的方法。

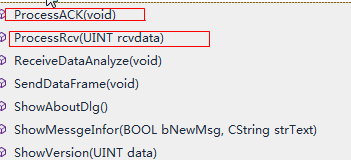
其他类似。



## 应答和接收

ProcessACK：不管读取和写入，每个指令发送后，软件都需要等待接收到一个ACK指令，表示你发送的数据它收到了。然后才能发送第二条信息

ProcessRcv：接收串口数据处理方法。其由一个定时器不断的调用。



# 20170708的一个修改实例

## 背景：

界面上有个四个数据表每个表示最大容量256，软件内部给每个表在Card\_Flash\_Table中分配了256个位置。一个表读取数据的完成后，界面显示都是顺序查询这个表在Card\_Flash\_Table表中对应的256个位置，有数据就显示到界面上，无数据就不显示。

## 王佳星需求：

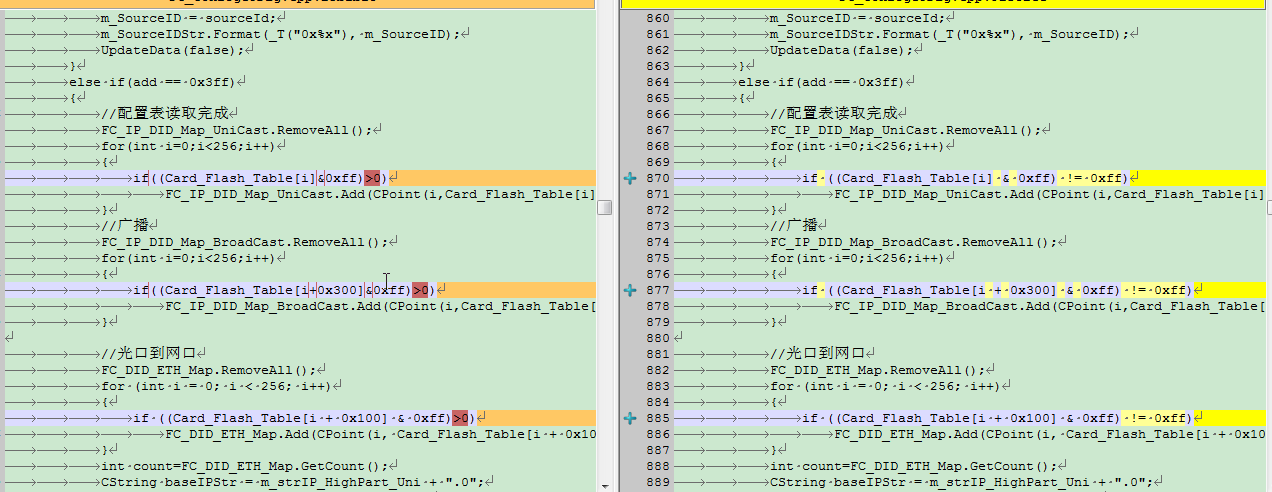
需求1：有数据用非0标识，无数据用0标识

需求2：无数据用FF标识，其他标识有数据

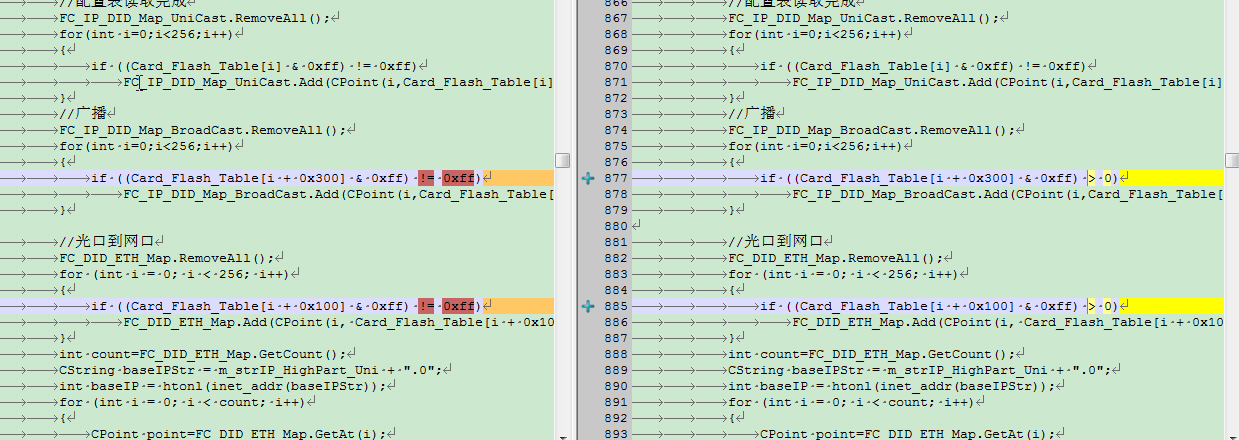
现在需求3：地址0~ff的以ff标识无效数据，其他以0标识无效数据。

## 根据需求代码修改的截图历史如下：

需求1—》 需求2：



需求2 – 》需求3：



具体情况可以使用git查看各个文件的修改详细情况

Over：是不是比较简单。