FF混合控制方案

实现思路

# 硬件配置模块生成链接关系

## 创建下装数据



Control->Link->Gateway->Slave

通过调用控制器的创建下装数据，调用CreateCfgModule创建数据，之后调用WriteDownloadmemFile下装数据

FF协议中CreateProtocolDownloadData，数据存放在CGeneralSlave::m\_pCfgModule->pSpecialData或

CGeneralLink::m\_pCfgModule->pSpecialData中

## 数据源

数据源是m\_LinkFBApplication，通过链路（主站）对象pLinkObj->GetItemValue(STR\_PROTOCOL\_DATA)获取

数据源定义：

m\_LinkFBApplication

1:n

FBApplication

1:n

FBAppDeviceInfo(m\_DeviceInfoMap)

----1:n

CONNECT\_SUB\_LIST(m\_connectMap)

1:n

CONNECT(uiPubChannelID、uiSubDeviceID、usSubBlockIndex、uiSubChannelID)

----1:n

LINKPUB(m\_listLinkPub) 连接到其它设备，自己加pub，其它加sub

----1:n

LINKTOVFB(m\_listLinkInternal)设备内部连接

----1:n

LINKSUB(m\_listLinkSub)

## 结论

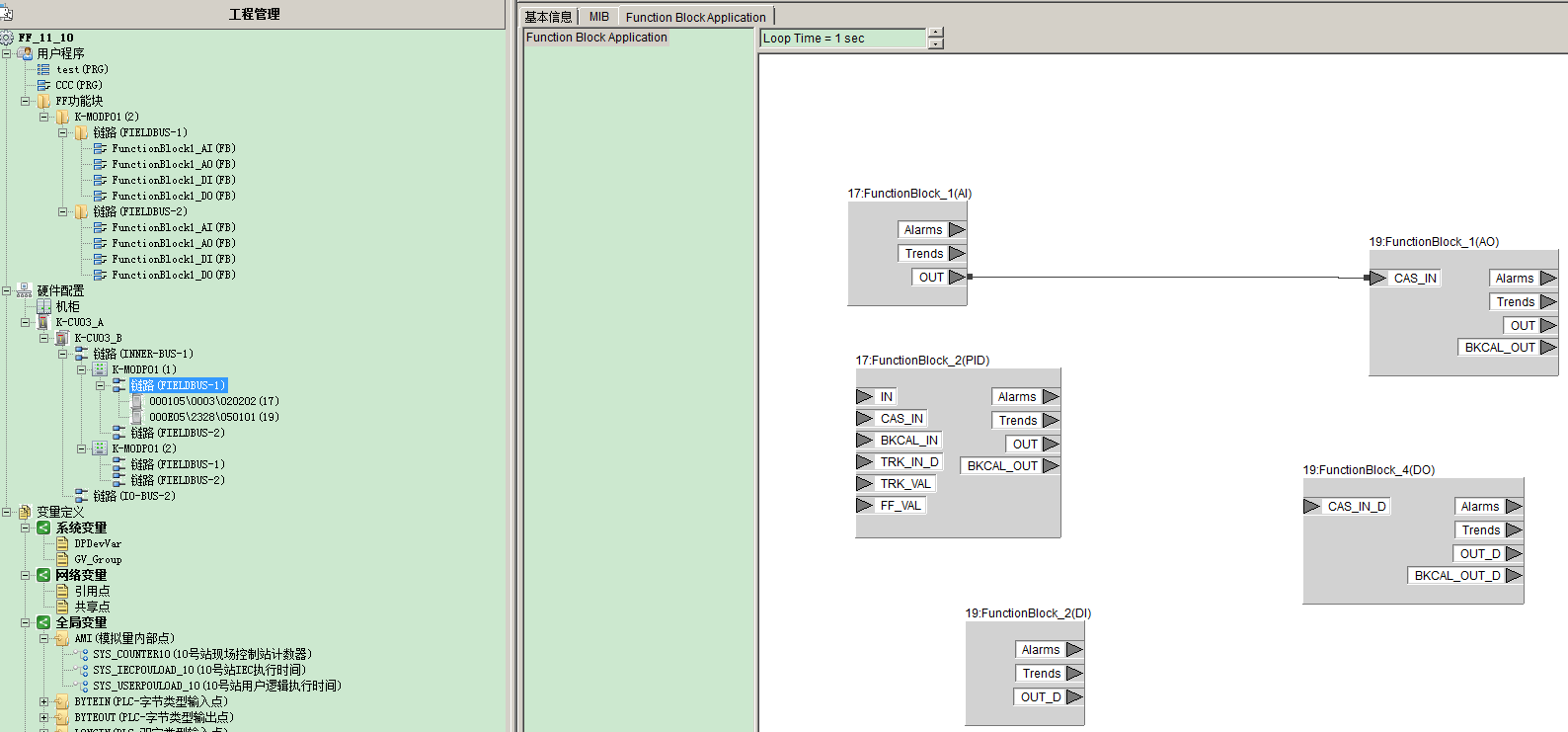
1. 数据源切换（函数块应用）

从站的数据源切换，还有主站的虚拟功能块支持

1. 下装的数据来源、意义

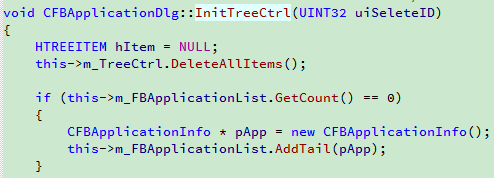
# 从站数据源切换

## Function Block Application构建



### 新建工程





初始装，strData为空，此时m\_FBApplicationList为空，在执行InitTreeCtrl时，会自动创建一个空的FBApplication加入到m\_FBApplicationList，在ChangeFBApp中赋值当前应用。

### 添加函数块

如下图，右键菜单，将Link子设备索引作为ucMainMenuID，子设备模块索引作为ucSubMenuID，拼凑为菜单ID



响应函数里，解析设备索引和模块索引



目的是构造出FBApplication的设备信息和块信息

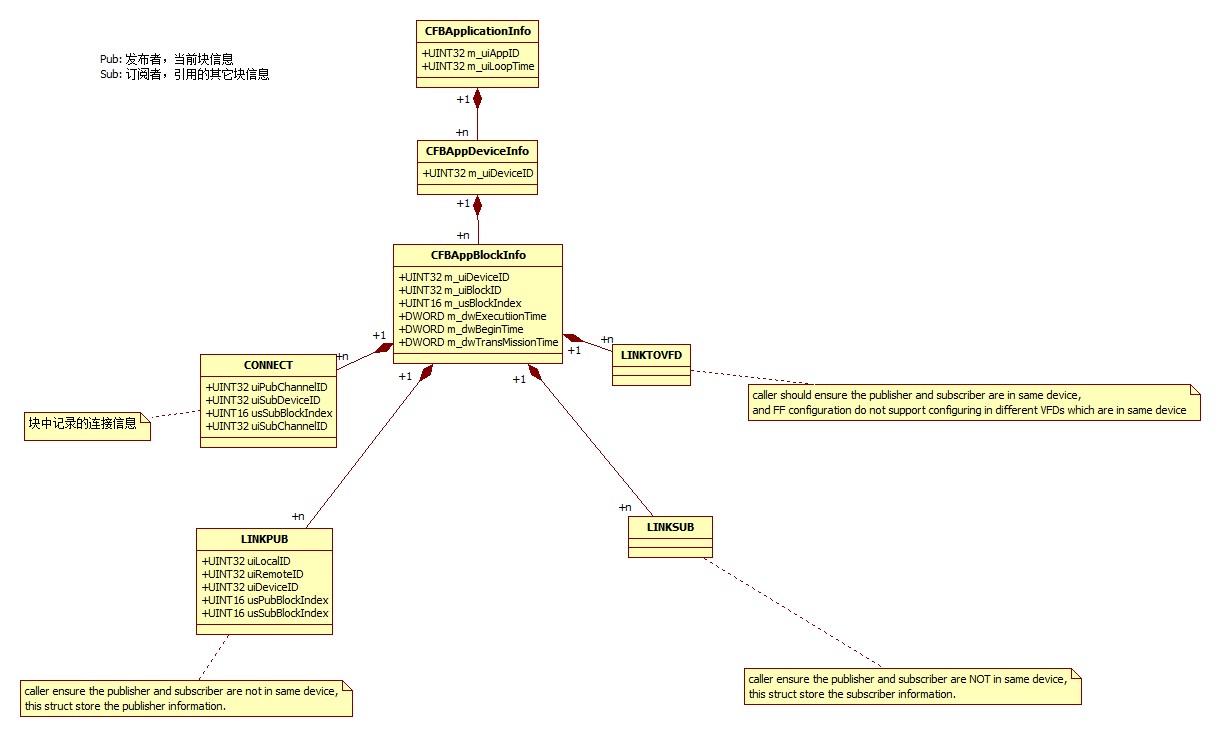


核心信息：链路下设备索引和模块索引

## 数据生成流程

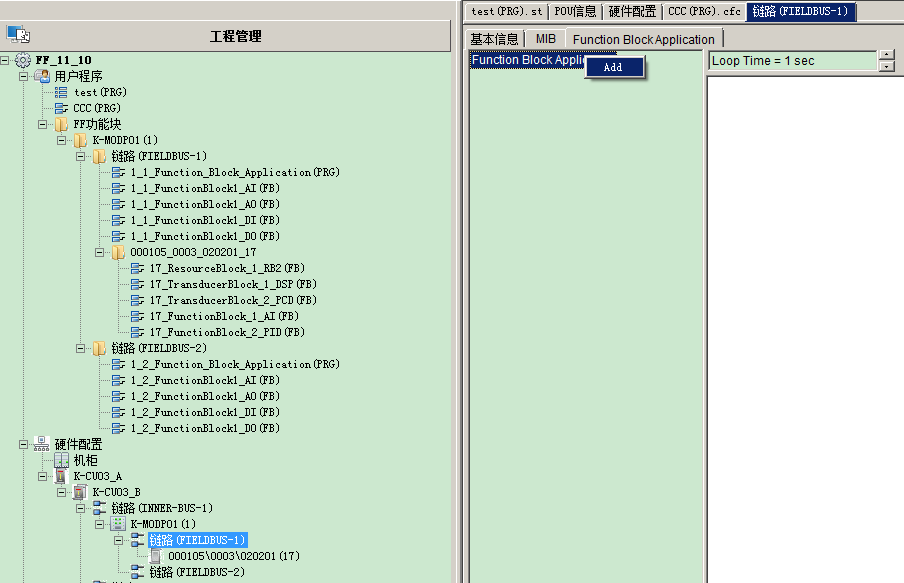
### 数据结构

如下：



### CFBAppliactionInfo生成

双击Link创建初始CFBAppliactionInfo，或在树有右键新增添加



### CFBAppDeviceInfo生成

说明：

设备信息

添加：

右键菜单添加设备时，填充m\_DeviceInfoMap



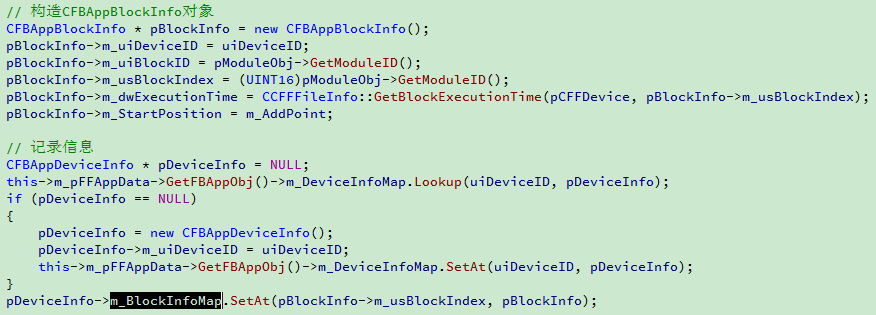
### CFBAppBlockInfo生成

说明：

应用块信息

添加：

生成CFBAppDeviceInfo时，生成对应的BlockInfo，目前逻辑，虽然关系是1:n，但实际是1:1



### CONNECT生成

说明：

被引用信息，即当前块指定引脚（发布者）关联的其它块的引脚信息（订阅者）

添加：

在鼠标拖动完毕时添加



### LINKPUB生成

说明：

无

添加：

不用主动添加，在下装时根据现有数据生成，见：CFBApplicationDlg::UpdateLinkageByConect

### LINKSUB生成

说明：

无

添加：

不用主动添加，在下装时根据现有数据生成，见：CFBApplicationDlg::UpdateBlockLinkSub

### LINKTOVFD生成

说明：

无

添加：

不用主动添加，在下装时根据现有数据生成，见：CFBApplicationDlg::UpdateLinkageByConect

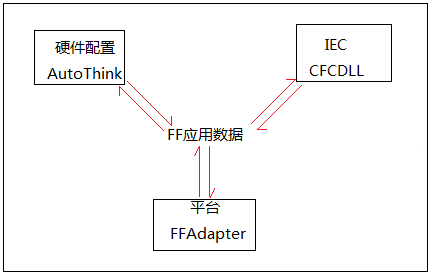
### 结论

关注数据有效性和添加方式

|  |  |  |
| --- | --- | --- |
| **数据** | **添加方式** | |
| **Function Block Application** | **IEC** |
| CFBAppliactionInfo | 双击Link或右键树添加 | 双击Link = 创建POU时自动生成 右键树添加 = 暂无方法，待考虑 |
| CFBAppDeviceInfo | 添加设备时添加 | 拖拽函数块添加 |
| CFBAppBlockInfo | 添加设备时添加 | 拖拽函数块添加 |
| CONNECT | 拖拽连线时添加 | 拖拽连线时添加 |
| LINKPUB | 下装时自动生成 | 无 |
| LINKSUB | 下装时自动生成 | 无 |
| LINKTOVFD | 下装时自动生成 | 无 |

## 数据切换

当前FF应用数据，在平台FFApdpter中，抽提后，需要保证硬件配置、IEC、平台都可以访问，首先在AT硬件配置操作添加主站、从站，FFAdapter将消息发送至框架，在框架里实现FF文件夹和POU、FB的创建，同时生成FF应用数据，之后在IEC操作中，填充FF设备信息、FF块信息、FF连接等信息，见下图：



### FF应用数据切换

#### 添加方式1

新建设备，打开链路，初始化FF应用对话框添加

硬件配置添加设备后，自动创建POU，

FBAppInfo初始添加

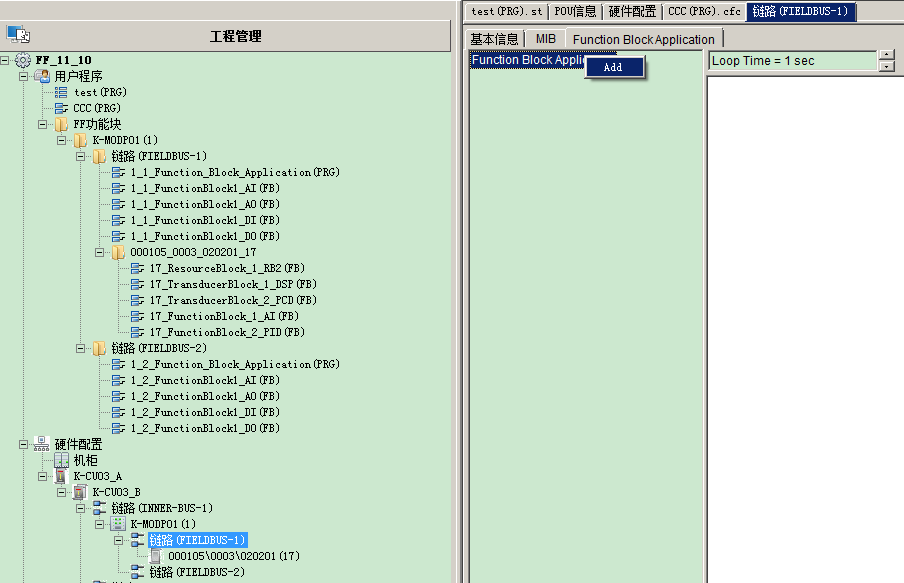




结论：支持，创建主站文件夹时，自动创建

#### 添加方式2

FBAppInfo新增，可以手动新增，即主站下除了默认的FF函数应用，还可以再加



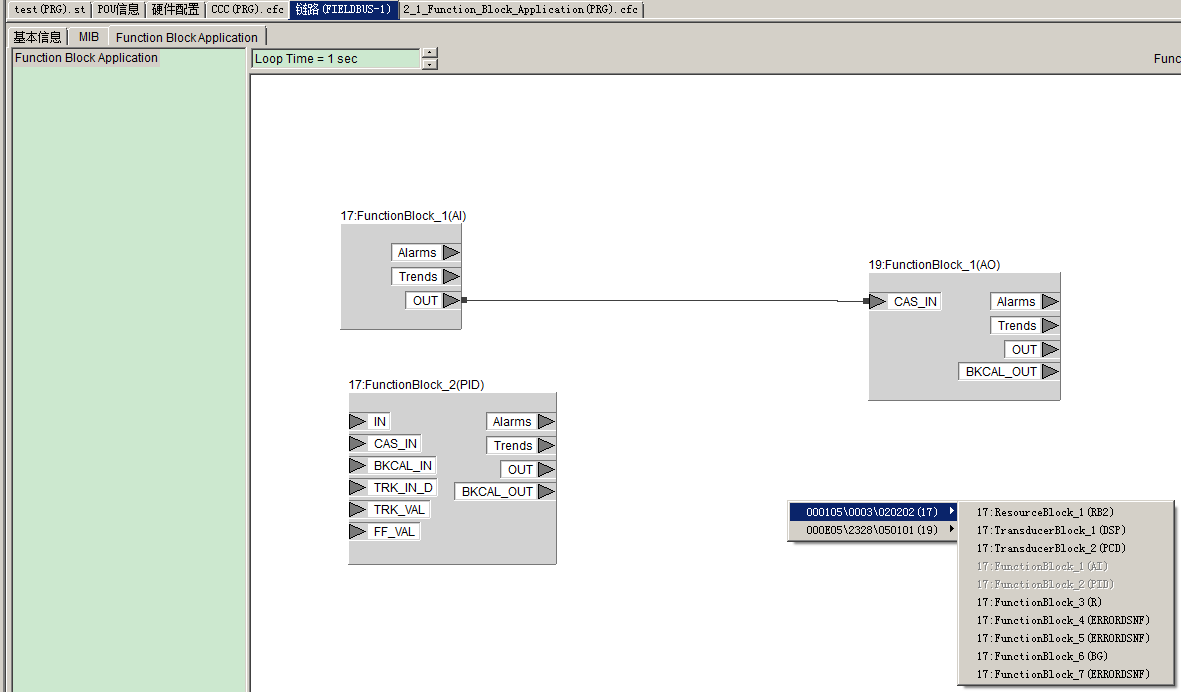


结论：数据切换，暂不支持

### FF设备数据切换

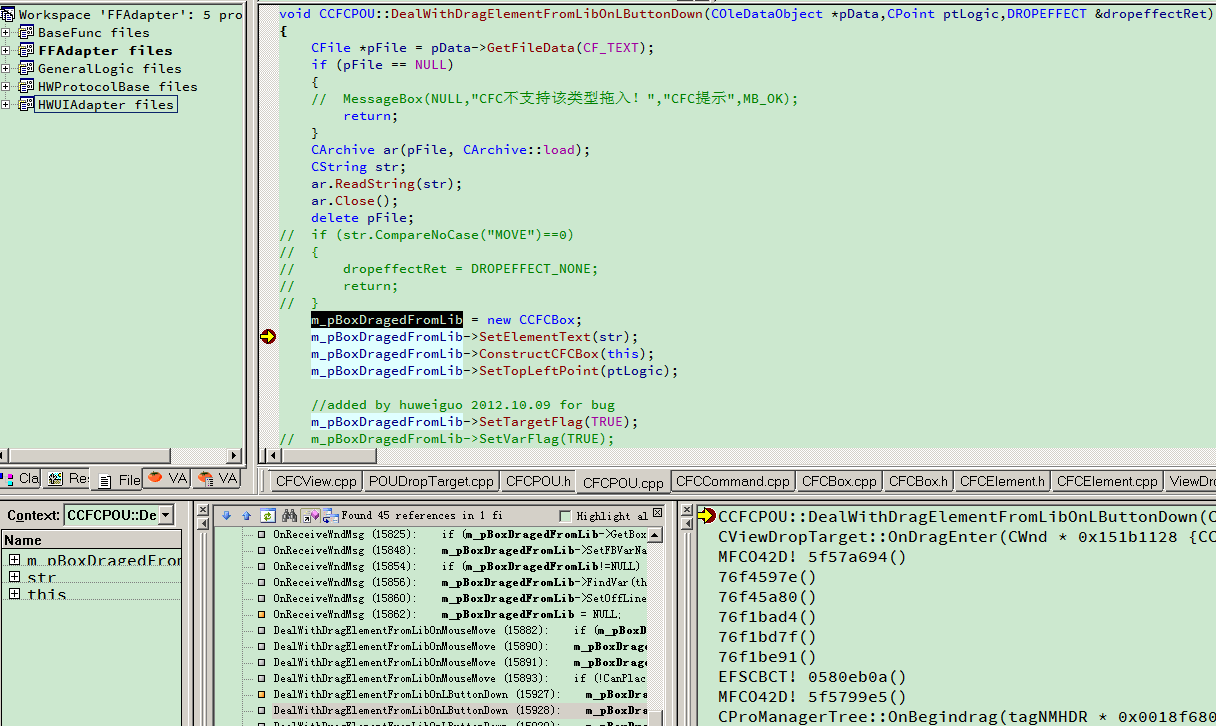
#### 添加方式

右键菜单添加



结论：

1. CFC中通过拖拽添加，拖拽处理函数



1. 拖拽实现



这里需要调整CFCFFInfo的作用域，之前是只AutoThink调用，现在IEC也调用，将该数据结构的定义实现为类，并提为公共类

1. 控制设备数据只能拖入1次



若已包含FF的设备块，则不能再拖入

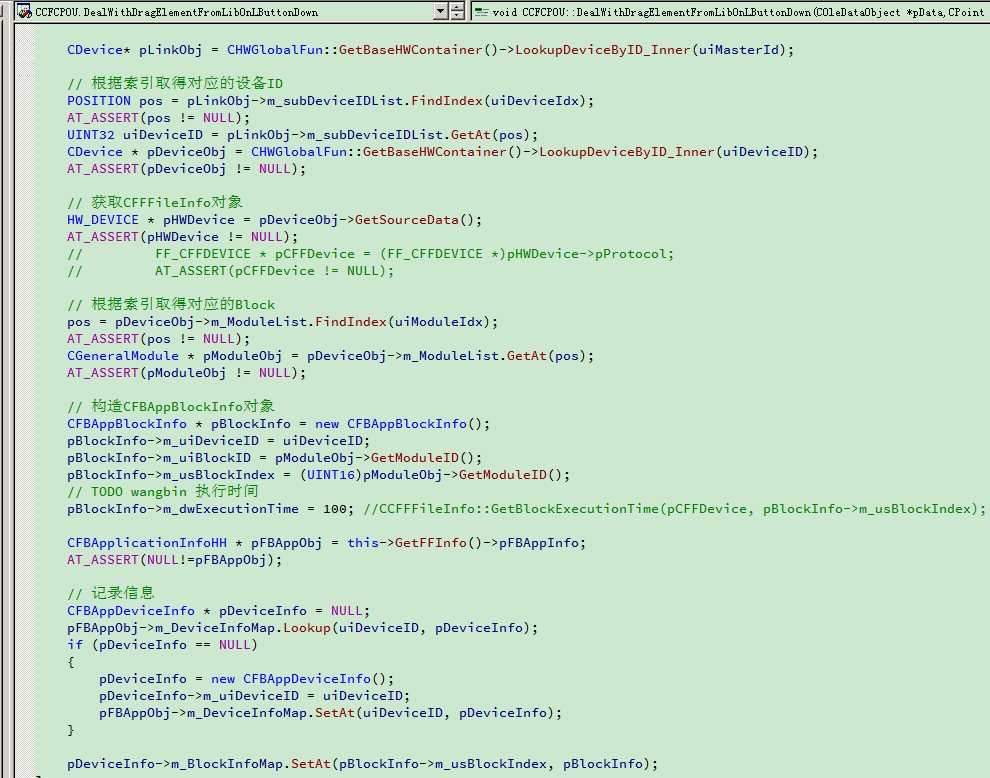
### FF块数据切换

#### 添加方式

同FF设备数据切换添加方式，在添加设备时，1对1生成设备数据信息和块数据信息

结论：

在拖拽CFC函数块时，生成对应的设备数据和块信息数据



#### 删除方式

在删除CFCBox块时，判断是否FF功能块



### FF连接数据切换

#### 添加方式

拉线

点下鼠标左键



放开鼠标左键

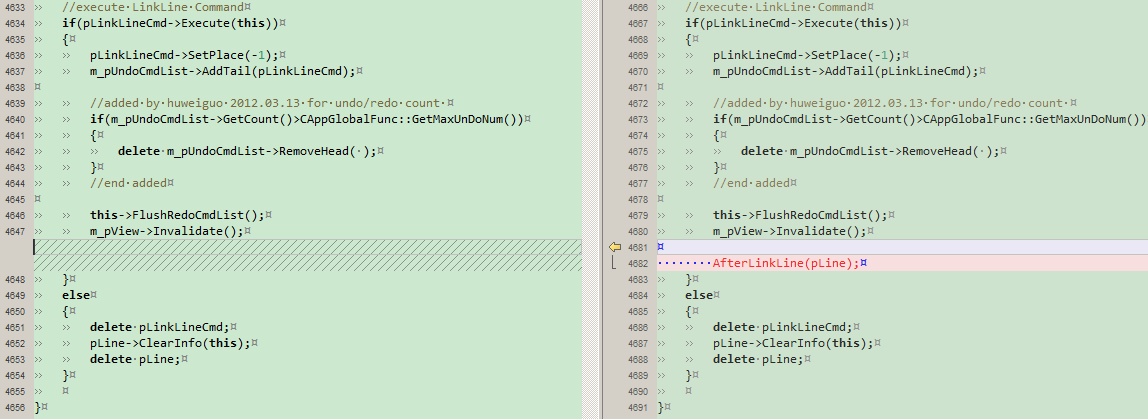


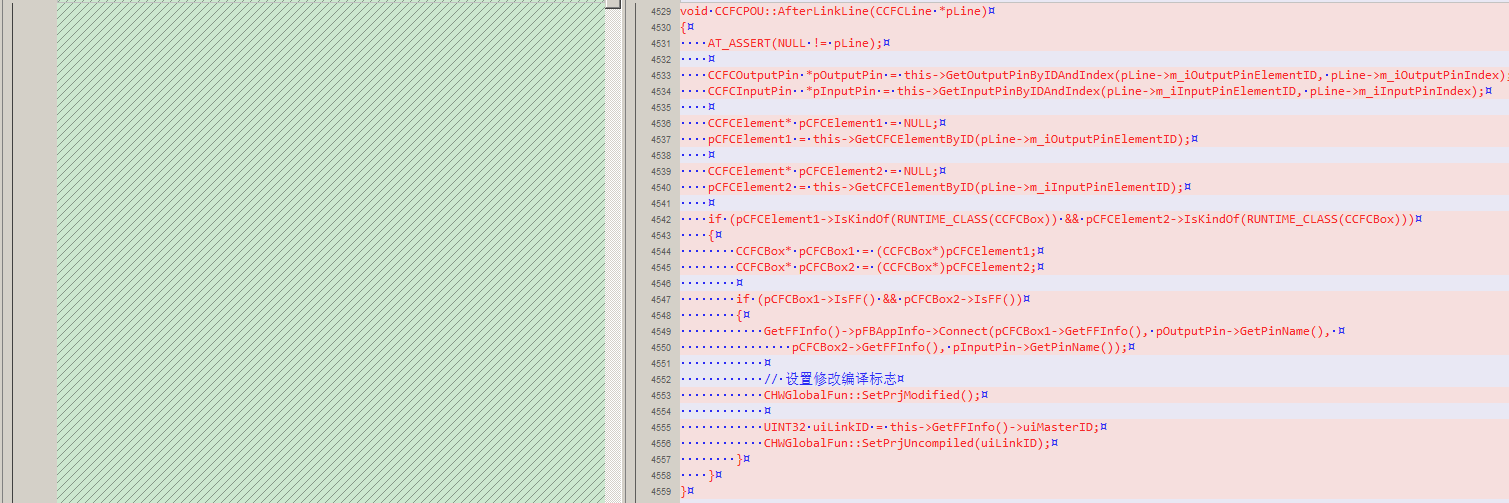
结论：

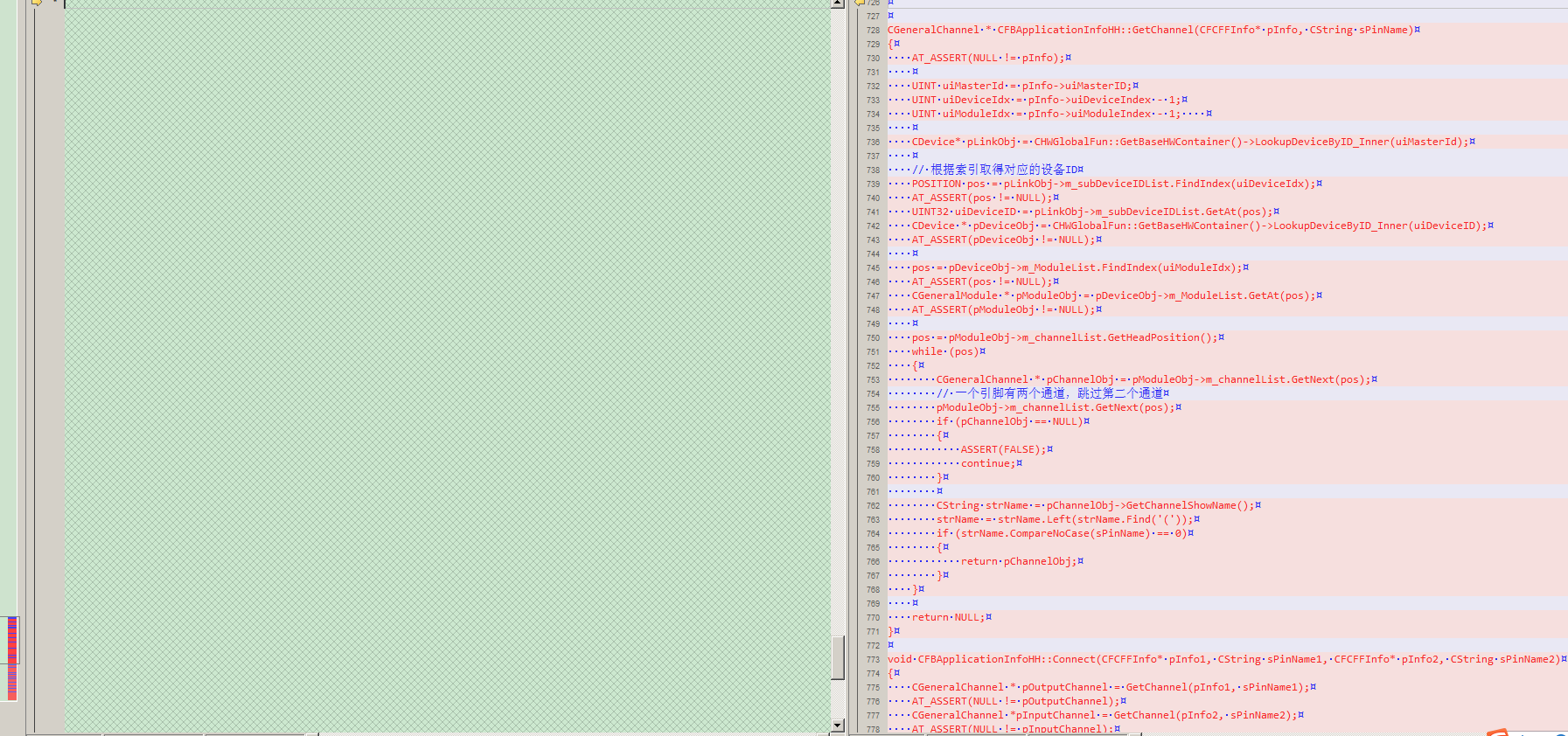
模拟FF连接模式，在CFC连接时，创建连接点



实现







# 虚拟主站功能块

## 初步实现计划

1. 根据CFF，根据从站生成虚拟主站

目前比较明确 ← 程工约1~2d完成

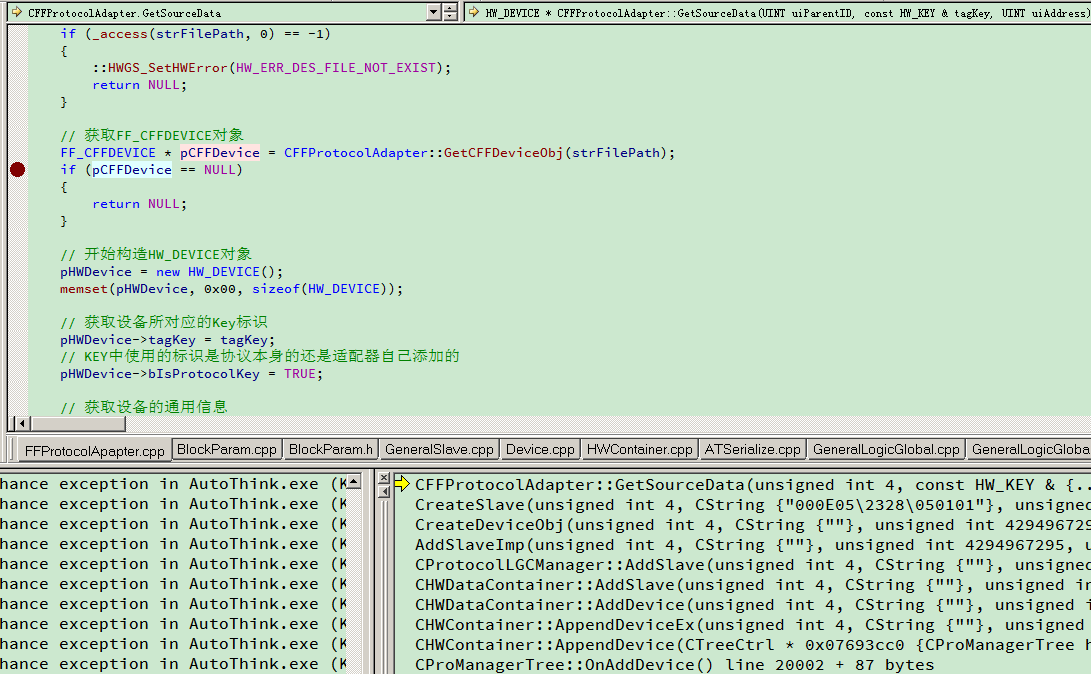
1. 构造虚拟主站的参数信息和通道信息

暂无方案，从站通过解析DD文件构造参数信息和通道信息，主站无DD文件

1. 需要预研下方案，如何构造主站信息 ← 王斌2~3d研究出个方案
2. 方案出来后，具体事项 ← 程工事项，时间待定
3. FF从站功能块和DCS混合组态，生成虚拟链接关系表 ← 王斌研究并实现部分
4. 虚拟链接关系如何对应
5. DCS算法块逻辑如何体现
6. FF、DCS通讯
7. 组态数据流转 ← 程工实现
8. 调度表
9. 各从站组态信息
10. 联调 ← 程工实现

## 构造虚拟主站的参数信息和通道信息

### 从站解析DD文件流程



经研究，创建从站时，在解析CFF文件中，由CFF指定读取位置从ffo文件中解析出参数信息和通道信息，最后生成FF\_CFFDEVICE数据，所以核心在FF\_CFFDEVICE的生成。

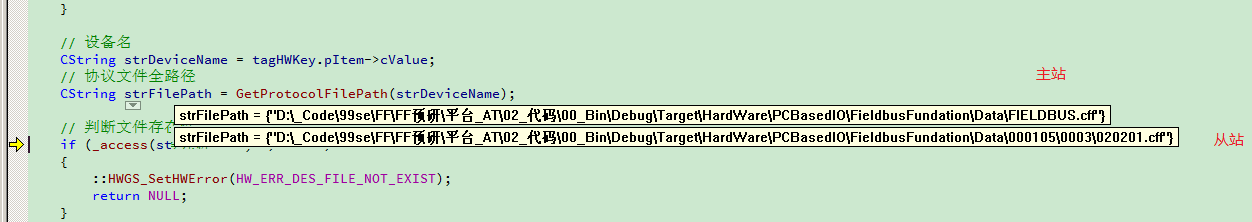
所以：

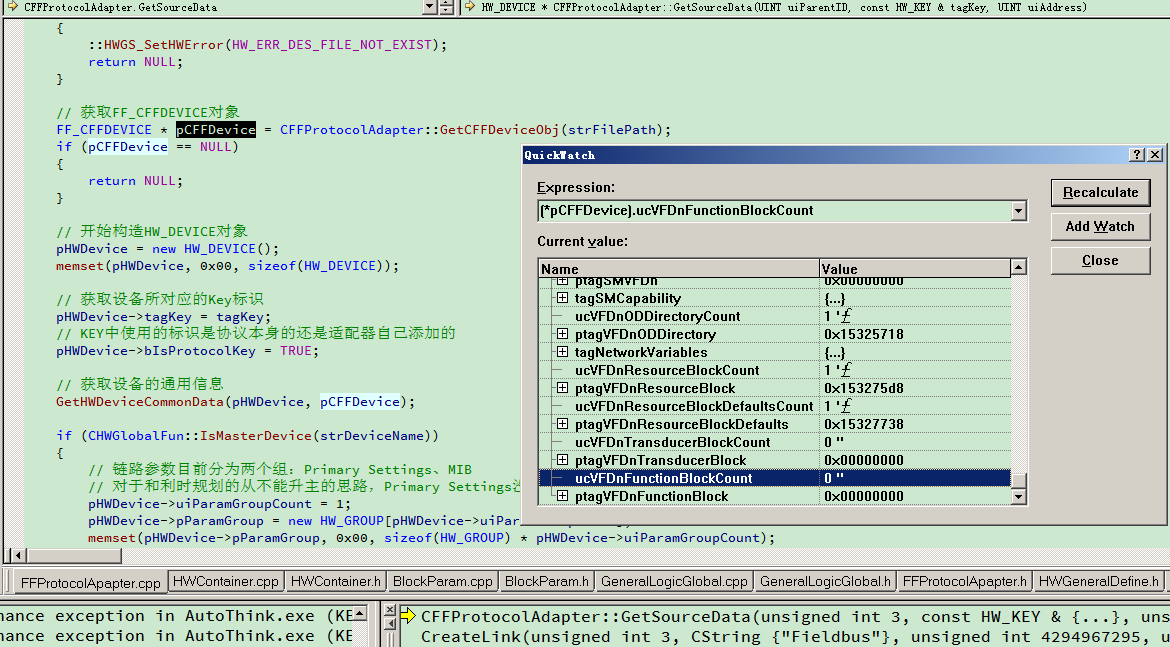
可尝试下面这个方案的可行性：

即在创建主站时，手动创建FF\_CFFDEVICE数据，并赋给主站，数据先写死，参考从站的数据

### 创建主站流程

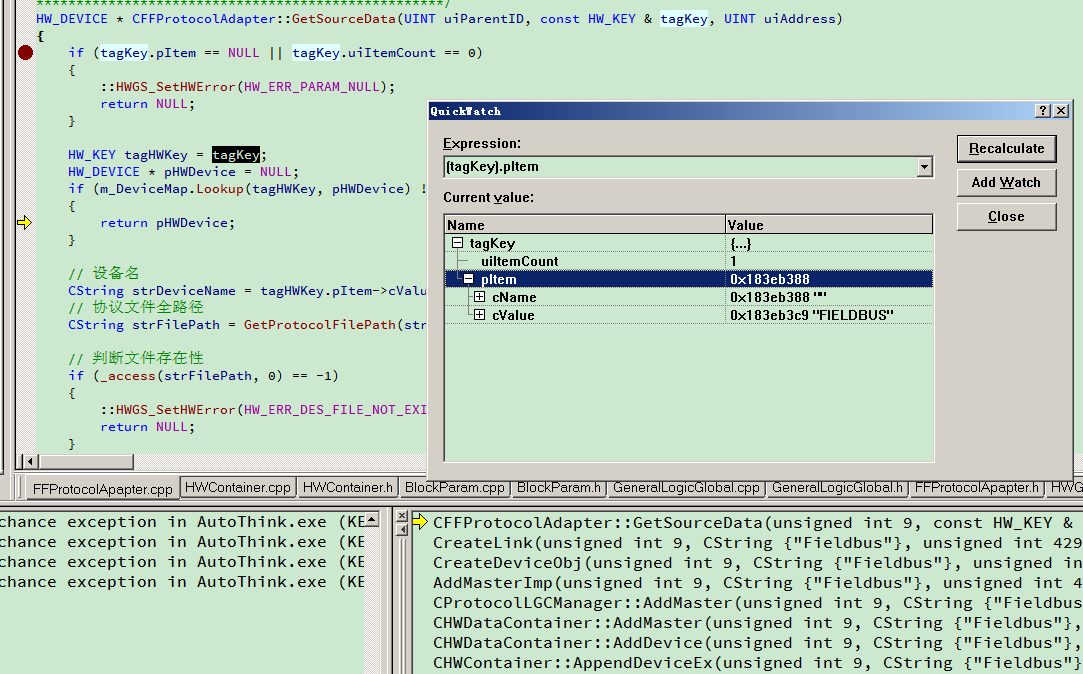
新建工程，首次添加主站（通过添加网关MODP-01），构造FF\_CFFDEVICE如下：





可以看出，读取CFF文件中个参数配置信息均为0

再次创建主站时，从m\_DeviceMap获取，不再读取CFF文件



结论：

1. 主站读取Fieldbus.cff获取配置信息，目前该配置信息中各数据为空
2. 从站读取cff、sym、ffo等文件获取配置信息

#### 创建方法

方法1：

仿从站CFFProtocolAdapter::GetCFFDeviceObj的实现，从数据层面分析从站的数据意义进行构造主站的数据

方法2：

CFF文件进行基本定义，可以模仿从站构造主站的

Sys和ffo是初始化块信息的，这2个文件是二进制，无法解析，只能分析从站数据进行构造

建议使用方法1，完全自己构造，有更大的自主权

#### 数据分析

通过解析文件cff、sys、ffo构造HW\_DEVICE信息

其中：

cff：描述设备信息

sys：符号数据，参数信息索引

ffo：参数信息（二进制文件）

HW\_DEVICE数据结构：

class HWProtocolBase\_DLL\_DECL HW\_DEVICE

{

public:

HW\_KEY tagKey;

// 参考HW\_PARAM该字段定义说明

BOOL bIsProtocolKey;

// 设备的共同数据部分定义

HW\_DEVICE\_COMMON tagCommon;

// 参数的组信息定义

UINT uiParamGroupCount;

HW\_GROUP \* pParamGroup;

// 设备的参数部分定义

UINT uiParamCount;

HW\_PARAM \* pParam;

// 模块的组信息定义

UINT uiModuleGroupCount;

HW\_GROUP \* pModuleGroup;

// 设备的模块部分定义

UINT uiModuleCount;

HW\_MODULE \* pModule;

// 协议私有部分定义

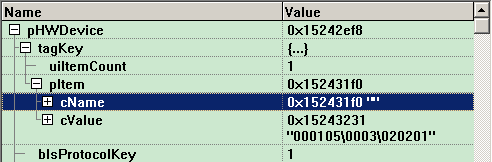
void \* pProtocol;

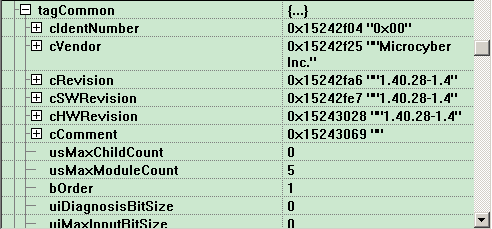
};

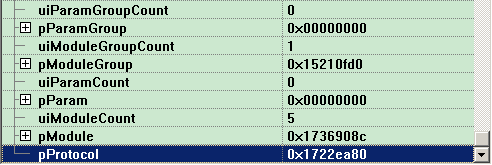
如上所示，核心在构造HW\_DEVICE下HW\_GROUP、HW\_PARAM、HW\_MODULE的数据，在根据从站数据构造主站数据时，可根据主站实际清楚决定初始化哪些数据

#### 数据示例

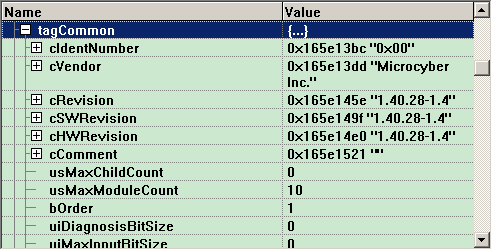
添加020201数据

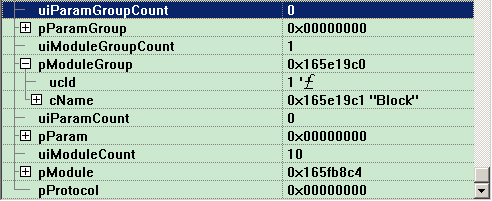


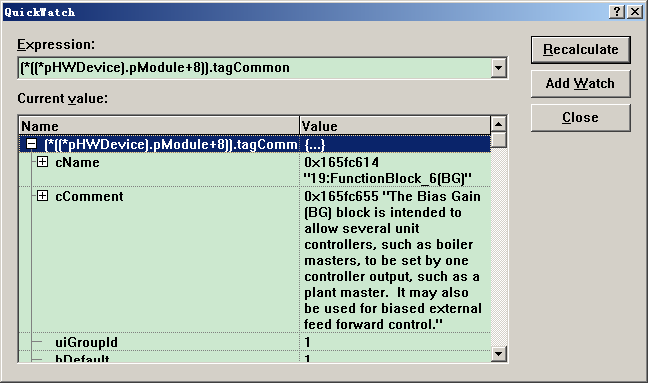


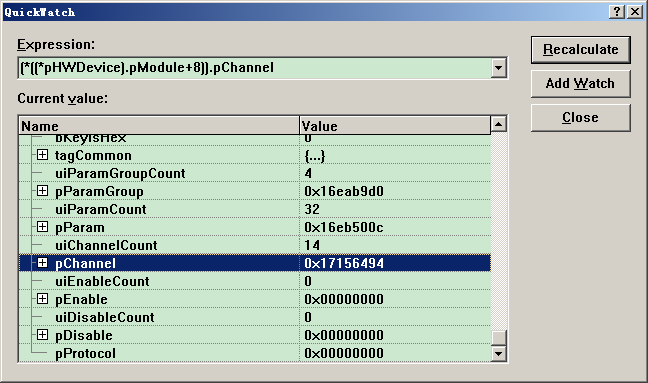


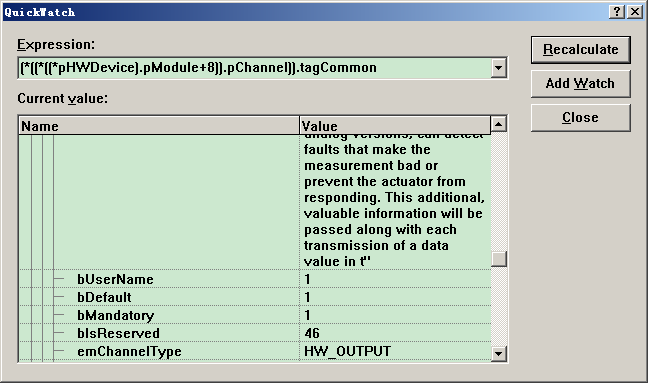
添加020202数据











结论：

HW\_DEVICE下的ParamGroup、Param信息均为空，ModuleGroup数量为1，Module数量若干，核心在构件Module的数据

结合主站特点，可以暂不关注Module的参数数据，先添加Channel数据即可

## 主站混合控制

### 虚拟连接关系表

#### 几个疑问

问题一：每个引脚2个通道：STATUS、VALUE

计算机生成了可选文字:
阜 120 0 
17 FLI n 0 n 0 ， 1 〔 司 ： 
OUT(STATUS) 
OUT(VALIJE) 

那主站24\*24引脚，实际对应的是48\*48个通道？

答：郝工那边只关心一个数据，这个数据包含状态和值，用户那边只关心引脚

所以可以和赵工沟通下，应该可以和从站一样（已和赵工沟通，先按从站处理）

问题二：主站通道地址，具体指啥？

（分析虚拟功能块可用的索引范围）

答：就是指前面的父索引值，先不用关注具体意义，主站内部唯一即可，可参照中控的索引范围用着（需要看下中控的范围值是多少）

具体有无规定，如AI必须是多少开始等，后期慢慢确定

这个值AT暂无法得出，需要分析中控下装数据通讯时的索引值（得找郝工协助）

问题三：FF从站之间是通过VCR通讯的，那FF设备和DCS如何通讯（DCS算法应该是个二进制数据）

答：RTS双口RAM

PowerLink主站

交互区域

需要维护一张表，主站索引&DCS模块地址的映射

在RTS中，拿到主站索引后，判断是混合控制的，直接从对应模块地址取值即可

所以这个的核心在

1. 对应表 2. RTS对混合控制的支持

DCS数据都存在RTS中

#### 虚拟关系表

* 数据如何在虚拟主站与RTS之间交互
* 通讯原理

       FF混合控制方案下，DCS算法块会虚拟一个主站功能块，这个功能块可以认为是一个实际的功能模块，它的通道地址和DCS变量地址的**映射表类似于硬件配置IO映射表**

       即：通讯时，可以利用这个IO映射表，添加虚拟主站功能块通道地址、DCS变量地址、读取（或写入）数据类型等信息；RTS在调度周期内通过映射表填充各地址信息供协议栈使用

**输入数据是由通道地址里的值交换到DCS变量地址里；输出数据是由DCS变量地址里交换到模块通道地址里**

* 映射表添加FF虚拟映射关系时机

       硬件配置编译保存

* 遗留问题

1. 直接利用IO映射表，当前方案是否支持（还是需要另建关系表？）

答：已和李鑫确认，扩大映射表的容量，当前是支持的，可以直接用

* 数据如何在协议栈与RTS之间交互

协议栈需要的数据存在RTS中，可通过通道地址直接从RTS里面取值

#### IO映射表

### 下发数据

#### 下发次序

数据层次：

主站

|--从1

|--从2

下发次序：

主站

主作从数据（需手动调用发送）

从1

从2

#### 下发数据

见下发次序，给协议栈下发的数据主要有两种，主站数据和从站数据（包含主作从数据）

* 主站下发的数据

保持现在给协议栈的数据不变，包括数据头和调度表，其中调度表信息包含从站和主作从的调度信息

* 从站下发的数据

保持现在给协议栈的数据不变，添加主作从，下发的各从站数据包含这几个从站的信息