比较部分逻辑

实现思路

# 实现背景

FF协议下装时需要判断参数实际值和默认值，若不一样才下装，否则不下装

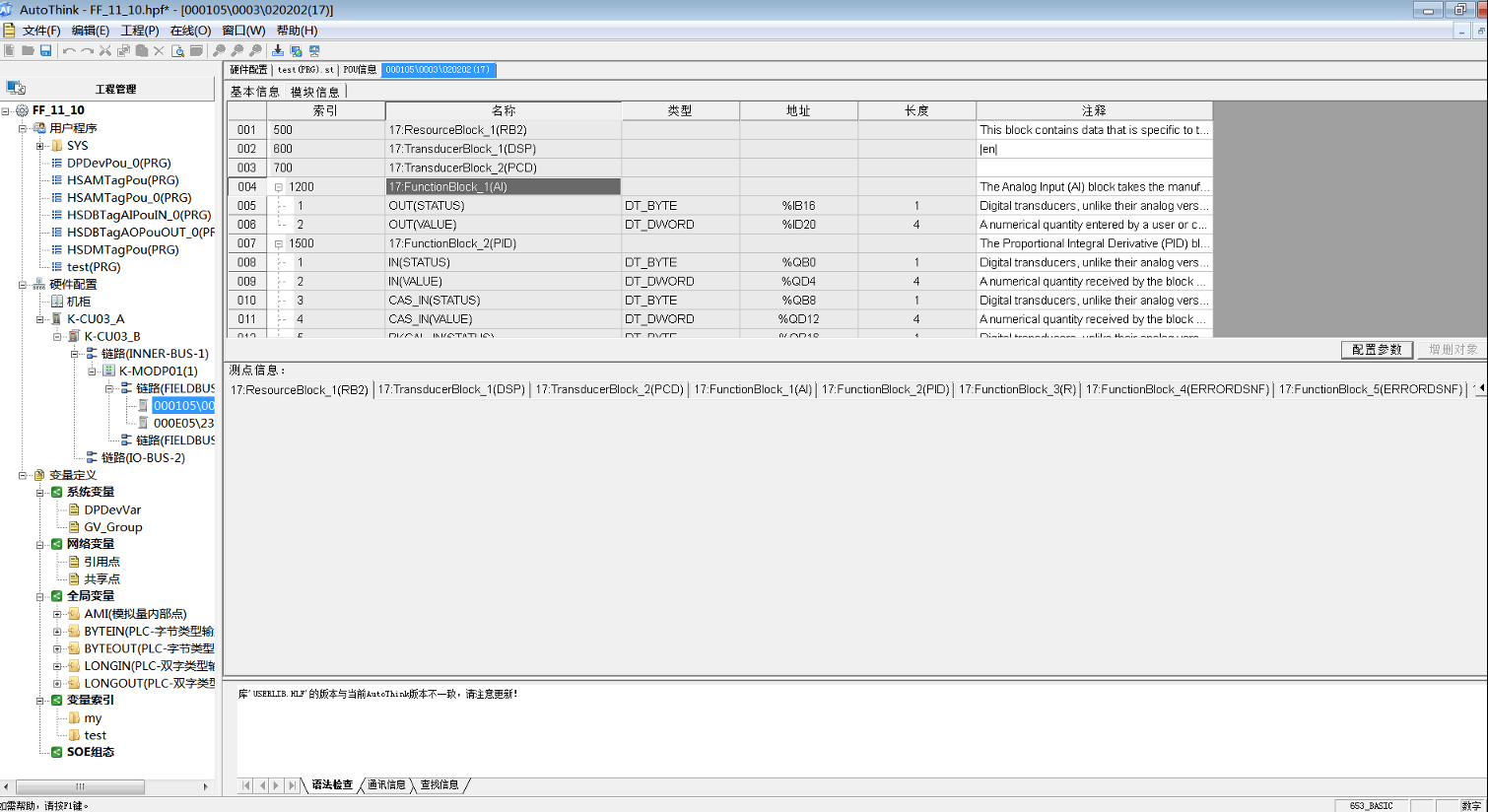
# 下装

## 入口函数

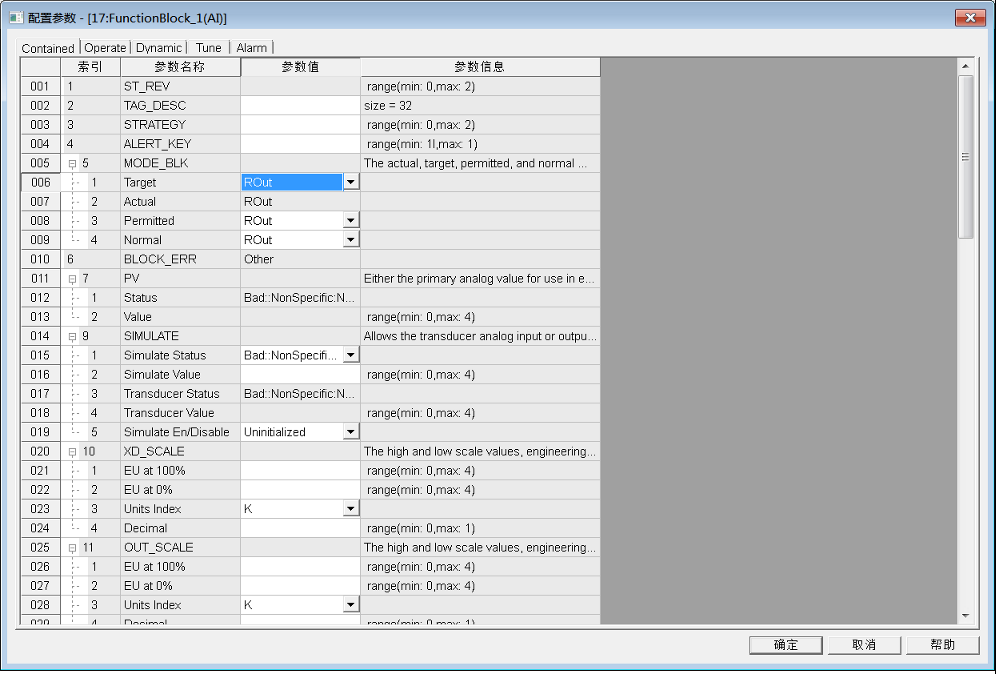
CAutoThinkApp::OnOnlineDownload

## 数据修改

新建工程，添加K-CU03控制站，在链路1下添加设备K-MODP01，在链路1下添加000105\0003\020202为地址17号设备，添加000E05\2328\050101为地址19号设备，如下图：



双击00105\0003\020202(17)，选中模块信息，选中17:FunctionBlock\_1(AI)，单击配置参数，如下图：



修改属性后，确定。

## 配置参数

界面CMultiTabDialog::OnOK

## 数据修改

# 修改思路

修改原则：

1. 属新增内容，在一堆的write后写入修改的参数值
2. 不分初次和增量，第一次下装就写入

 修改位置：

CFFProtocolAdapter::CreateProtocolDownloadData

写入步骤：

1. 拿到所有功能块参数的信息
2. 拿到所有功能块参数的默认信息
3. 比较数据
4. 和郝丽定义好格式
5. 写入数据

# 参考实现

## 形成下装数据包

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*功能                : 形成下装数据包

\*参数                : 控制器ID，全数据、增量数据

\*返回值                : 无错误返回TRUE

\*创建者                : lzj

\*创建日期        :        2015.5.9

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

BOOL HWCreateDownloadDataImp(UINT32 uiControlID,CObject\* pControlLSTDownloadInfo,INT32 &iSizeAll,char\*& pBufAll, INT32& iSizeAdd,char\*& pBufAdd)

{

CControl\* pControl = NULL;

pControl = (CControl\*)CHWGlobalFun::GetBaseHWContainer()->LookupDeviceByID\_Inner(uiControlID);

if (NULL != pControl)

{

return pControl->HWCreateDownloadData(pControlLSTDownloadInfo,iSizeAll,pBufAll,iSizeAdd,pBufAdd);

}

else

{

iSizeAll = 0;

iSizeAdd = 0;

return TRUE;

}

}

## 取得需要下装的参数个数

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*功能                : 取得需要下装的参数个数

\*参数                : 无

\*返回值                : 需要下装的参数个数

\*创建者                : Zhao Liping

\*创建日期        :        2016.08.26

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

UINT CGeneralSlave::GetDownloadParamCount()

{

UINT uiParamCount = 0;

HW\_DEVICE \* pDevice = this->GetSourceData();

if (pDevice == NULL)

{

::HWGS\_SetHWError(HW\_ERR\_GET\_SOURCE\_NULL);

return 0;

}

HW\_PARAM \* pParam = NULL;

UINT uiCount = pDevice->uiParamCount;

for (UINT uiIdx = 0; uiIdx < uiCount; ++uiIdx)

{

pParam = pDevice->pParam + uiIdx;

if (pParam->bIsDownload)

{

++uiParamCount;

}

}

return uiParamCount;

}

## 生成指定设备的下装数据接口

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*功能 : 生成指定设备的下装数据接口

\*参数 : pDevice: 设备对象

bLinkIsBigEnd:主站的大小端标志

pBuff: 返回下装数据内存

uiLen: 返回下装数据长度

\*返回值 : 成功返回TRUE，否则FALSE

\*创建者 : Zhao Liping

\*创建日期 : 2016.05.27

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

BOOL CFFProtocolAdapter::CreateProtocolDownloadData(

CDevice \* pDevice,

BOOL bLinkIsBigEnd,

BYTE \*& pBuff,

UINT32 & uiLen)

{

if (pDevice == NULL)

{

::HWGS\_SetHWError(HW\_ERR\_PARAM\_NULL);

return FALSE;

}

// 初始化pBuff

pBuff = NULL;

uiLen = 0;

CMemFile memFileObj;

// 头部信息

UINT32 uiVersion = 0x00000001;

UINT32 uiCheckSum = 0x00000000;

UINT32 uiLength = 0x00000000;

if (this->IsBigEnd())

{

uiVersion = BaseFunc::TransEndian32(uiVersion);

}

memFileObj.Write(&uiVersion, sizeof(uiVersion));

memFileObj.Write(&uiCheckSum, sizeof(uiCheckSum));

memFileObj.Write(&uiLength, sizeof(uiLength));

// 生成设备的下装信息，目前平台调用的生成下装特殊数据的顺序是先从再主，

// 所以在从站时要准备下主的信息，需要一些特殊处理

if (CHWGlobalFun::IsMasterDevice(pDevice->GetDeviceName()))

{ // 链路

// 写入静态VCR信息

// Write\_Vcr\_Static\_Objects(pDevice, memFileObj);

// 在链路的特殊数据区写入调度信息

Write\_Schedule\_Object(pDevice, memFileObj);

// 主站特殊数据创建完成后清空从站调用PrepareMasterDownload生成的信息

if (this->m\_TempMasterID != (UINT32)-1)

{

this->m\_TempMasterID = (UINT32)-1;

ClearMasterDownloadData();

}

else

{ // 这种清空下只能是主站下没从站的场合

AT\_ASSERT(pDevice->m\_subDeviceIDList.GetCount() == 0);

}

}

else

{ // 从站

// 特殊数据先从后主，一部分数据是主站准备，从站下装需要，所以有以下逻辑

if (this->m\_TempMasterID == (UINT32)-1)

{

this->m\_TempMasterID = pDevice->GetParentID();

PrepareMasterDownloadData(this->m\_TempMasterID);

}

// 从站下装数据格式

// UINT32 uiVersion

// UINT32 uiCheckSum // 计算是该字段置0，计算完后回填

// UINT32 uiLength // 总的长度，包括头部12个字节

// 对象字典项的下装顺序

// S\_OD\_ENTRY Nmib\_Object[MAXCHIP];

// S\_OD\_ENTRY Smib\_Object[MAXCHIP];

// S\_OD\_ENTRY sm\_t1[MAXCHIP];

// S\_OD\_ENTRY sm\_t2[MAXCHIP];

// S\_OD\_ENTRY sm\_t3[MAXCHIP];

// S\_OD\_ENTRY SM\_Local\_Time\_Diff[MAXCHIP];

// S\_OD\_ENTRY SM\_Ap\_Clock\_Sync\_Interval[MAXCHIP];

// S\_OD\_ENTRY SM\_Primary\_Ap\_Time\_Publisher[MAXCHIP];

// S\_OD\_ENTRY SM\_Macrocycle\_Duration[MAXCHIP];

// S\_OD\_ENTRY SM\_Vfd)Ref\_Entry\_Objects\_1 [MAXCHIP];

// S\_OD\_ENTRY SM\_Version\_Of\_Schedule [MAXCHIP];

// S\_OD\_ENTRY Vcrl\_Control\_Object [MAXCHIP];

// S\_OD\_ENTRY Vcr\_Static\_Objects [MAXCHIP][MAX\_VCR];

// S\_OD\_ENTRY Dl\_Basic\_Char\_Object[MAXCHIP];

// S\_OD\_ENTRY Dl\_Basic\_Info\_Object [MAXCHIP];

// S\_OD\_ENTRY Dl\_Link\_Capa\_Object [MAXCHIP];

// S\_OD\_ENTRY Primary\_Lm\_Flag\_Object[MAXCHIP];

// S\_OD\_ENTRY Max\_Token\_Holdtime\_Object[MAXCHIP];

// S\_OD\_ENTRY Boot\_Operation\_Object[MAXCHIP];

// S\_OD\_ENTRY Config\_Link\_Setting\_Object[MAXCHIP];

// S\_OD\_ENTRY Link\_Schedule\_Activation\_Object[MAXCHIP];

// S\_OD\_ENTRY Las\_List\_Object [MAXCHIP];

// 对象字典项信息

Write\_Nmib\_Object(pDevice, memFileObj);

Write\_Smib\_Object(pDevice, memFileObj);

Write\_sm\_t1(pDevice, memFileObj);

Write\_sm\_t2(pDevice, memFileObj);

Write\_sm\_t3(pDevice, memFileObj);

Write\_SM\_Local\_Time\_Diff(pDevice, memFileObj);

Write\_SM\_Ap\_Clock\_Sync\_Interval(pDevice, memFileObj);

Write\_SM\_Primary\_Ap\_Time\_Publisher(pDevice, memFileObj);

Write\_SM\_Macrocycle\_Duration(pDevice, memFileObj);

Write\_SM\_Vfd\_Ref\_Entry\_Objects\_1(pDevice, memFileObj);

Write\_SM\_Version\_Of\_Schedule(pDevice, memFileObj);

Write\_Vcrl\_Control\_Object(pDevice, memFileObj);

CDevice \* pLinkObj = CHWGlobalFun::GetBaseHWContainer()->LookupDeviceByID\_Inner(this->m\_TempMasterID);

Write\_Master\_Vcr\_Static\_Objects(pLinkObj, memFileObj);

Write\_Slave\_Vcr\_Static\_Objects(pDevice, memFileObj);

Write\_Start\_Block(pDevice, memFileObj);

Write\_Linkage(pDevice, memFileObj);

Write\_Mode\_BLK\_Index(pDevice, memFileObj);

Write\_CycleDataInfo(pDevice, memFileObj);

Write\_Dl\_Basic\_Char\_Object(pDevice, memFileObj);

Write\_Dl\_Basic\_Info\_Object(pDevice, memFileObj);

Write\_Dl\_Link\_Capa\_Object(pDevice, memFileObj);

Write\_Primary\_Lm\_Flag\_Object(pDevice, memFileObj);

Write\_Max\_Token\_Holdtime\_Object(pDevice, memFileObj);

Write\_Boot\_Operation\_Object(pDevice, memFileObj);

Write\_Config\_Link\_Setting\_Object(pDevice, memFileObj);

Write\_Link\_Schedule\_Activation\_Object(pDevice, memFileObj);

Write\_Las\_List\_Object(pDevice, memFileObj);

Write\_Dl\_Link\_Info\_Object(pDevice, memFileObj);

}

// 回填长度

uiLength = memFileObj.GetLength();

if (this->IsBigEnd())

{

uiLength = BaseFunc::TransEndian32(uiLength);

}

memFileObj.Seek(8, CFile::begin);

memFileObj.Write(&uiLength, sizeof(uiLength));

// 计算校验和，返回长度和内容

uiLen = memFileObj.GetLength();

pBuff = new BYTE[uiLen];

memFileObj.Seek(0, CFile::begin);

memFileObj.Read(pBuff, uiLen);

uiCheckSum = BaseFunc::CreateCRC32Checksum((char \*)pBuff, uiLen);

if (this->IsBigEnd())

{

uiCheckSum = BaseFunc::TransEndian32(uiCheckSum);

}

\*((UINT32 \*)(pBuff + 4)) = uiCheckSum;

memFileObj.Close();

CString strFileName = "";

strFileName.Format("D:\\Download(%d).dat", pDevice->GetID());

FILE \* fp = fopen(strFileName, "wb");

fwrite(pBuff, uiLen, 1, fp);

fclose(fp);

return TRUE;

}