硬件配置数据结构

-底园园 2014.8

1. 前言

硬件配置模块是AutoThink软件的重要组成部分，是现场硬件设备组态的逻辑映射，同时提供友好、简洁的组态界面，更方便用户快捷完成组态，更直观表现现场硬件组态方式。

硬件配置主要完成现场输入设备的码值采集，以及输出设备的信号下发。它提供和利时自主研发的硬件设备展示、极友好的组态界面，同时支持其它公司发布的标准数据文件以及友好方便的组态方式。

硬件配置模块支持硬件设备的无扰下装，能实现单个从站级的修改而不影响其他从站设备的运行。

1. 总体数据结构

根据数据的来源、显示及处理，硬件配置内部划分三个动态库管理数据，相互之间职责分明，任务清晰。

这三个动态库分别为：

1. 源数据：指GSD等配置文件对应的数据信息，它们是整个硬件配置的数据源泉；
2. 逻辑数据：储存主从站的逻辑信息，保存用户界面修改值，形成下装数据；
3. 显示数据：根据数据源中的配置信息和存储的逻辑数据进行显示。



图 1：硬件配置内部的数据依赖关系

1. 源数据结构

源数据的组成如下图所示。

图 2：源数据来源

包括以下几大类：

* 类CGsd：解析GSD文件并存储信息。

主要成员有：

* 解析的GSD详细信息

CMap<CString,LPCTSTR,DP\_DEVICE\_PARAM\*,DP\_DEVICE\_PARAM\*&> m\_gsdMap

* 存储GSD的基本信息

CMap<CString,LPCTSTR,DP\_BASE\_GSD\*,DP\_BASE\_GSD\*&> m\_gsdBaseMap

* 类CCFG：解析CFG文件并存储信息。

主要成员有：

* + - * 从AT2.cfg文件读取的基本信息

HWGlobalCfg\* m\_pGlobalCfg\_T

* + - * 从AT2.cfg文件读取的从站配置信息

CMap<CString,LPCTSTR,HWCardCfg\*,HWCardCfg\*&> m\_blockMap

* + - * 从AT2.cfg文件读取的控制器配置信息

CMap<CString,LPCTSTR,HWControlCfg\*,HWControlCfg\*&> m\_controlMap

* + - * 从AT2.cfg文件读取的字体信息

CMap<CString,LPCTSTR,FontDesc\*,FontDesc\*&> m\_fontMap

* + - * 从Cabinets.ini文件读取的机柜配置信息

CMap<CString,LPCTSTR,HWCabinetCfg\*,HWCabinetCfg\*&> m\_CabinetMap

* 类CHWDesc：解析得到设备库等信息。

主要成员有：

* 从HWLibTree.cfg读取的设备库信息

CMap<CString,LPCTSTR,TreeDesc\*,TreeDesc\*&> m\_itemMap

CList<TreeDesc\*,TreeDesc\*&> m\_itemList

* 从AT\_Configuration.ini文件中读出的主控与机柜的对应信息

CMap<CString, LPCTSTR, CArray<CString,CString>\*, CArray<CString,CString>\*&> m\_strConrol\_ShelfMap

* 类CHWDeviceInfos：解析得到所有的设备硬件属性信息。
* 类CHWParamLimit：解析得到设备参数及参数间约束关系。

主要成员：

* 存储HWParamLimit.ini的基本信息

CMap<CString,LPCTSTR, CString, CString> m\_HWParmLimitBaseMap

* 存储HWParamLimit.ini的详细信息

CMap<CString,LPCTSTR, HWParamLimit\*, HWParamLimit\*&> m\_HWParmLimitMap

以上类信息的对象存放在源数据容器CHWSourceContainer中。如下：



图 3：源数据容器

除了以上信息外，容器CHWSourceContainer还存放了以下信息：

* 从ModuleChannle.ini文件读取的各模块信息

CMap<CString,LPCTSTR,ModInfo\*,ModInfo\*&> m\_ModInfoMap

CList< ModInfo\*, ModInfo \*&> m\_ModInfoList

* 从GSD文件读取的各模块的MODULE信息

CMap<CString,LPCTSTR,SlaveGSDInfo\*,SlaveGSDInfo\*&> m\_SlaveGSDInfoMap

* 从Pointer.ini读出的测点类型与系统POU逻辑的对应关系

CMap<CString,LPCTSTR,PouAndTestPointInfo\*,PouAndTestPointInfo\*&> m\_PouAndTestPointInfoMap

* 从GroupSetting.ini读出的SOE模块的组设置信息

CMap<CString,LPCTSTR,SlaveGrpInfo\*,SlaveGrpInfo\*&> m\_SlaveGrpInfoMap

1. 逻辑数据结构

逻辑数据DLL实现控制器的添加、模块的添加和删除、逻辑数据组态错误检查及硬件配置下装数据包生成等功能。

类结构图：



图 4：逻辑数据的类结构图

* 类CDataContainer：存储组态的逻辑数据容器。组态的控制器及模块均存放在数据容器类CHWDataContainer中。

主要成员：

* CMap<UINT,UINT,CDevice\*,CDevice\*&> m\_elementMap是以元素ID号为标识、逻辑对象指针为成员的Map表，这里的逻辑对象是指控制器和模块对应的逻辑数据。
* CControl\* m\_pControl 是控制器对象的指针。
* 类CControl：控制器设备。

主要成员：

CList<CDPLink\*,CDPLink\*&> m\_linkList 表示当前控制器的链路列表，目前为单链路设计。

* 类CDPLink：主站下挂链路。

主要成员：

* + - char\* m\_pDownloadData 表示该链路的下装数据。
    - unsigned char m\_ucSlaveStateParam[16] 表示该链路下所有从站设备下装时刻的变化标记，用于支持从站的增量下装。

每字节按从低位到高位的顺序，依次表示各站号对应的从站是否变化，若变化，则该位为0，若不变化，则该位为1。

* 类CSlave：从站设备。

主要成员：

* BYTE m\_uchAddress 表示从站设备的地址。
* DP\_DEVICE\_PARAM \* m\_pSlaveParam\_t 表示该设备对应的GSD源数据指针。
* UCHAR\* m\_pDownloadData表示该设备的下装数据。
* CList<CModule\*,CModule\*&> m\_moduleList表示从站设备的子模块列表
* CList<CChannel\*,CChannel\*&> m\_channelList表示从站设备的通道列表
* 类CModule：从站设备下配置的子模块信息。

主要成员：

* CString m\_strName 表示该子模块的名字。
* UINT m\_uiSlaveID表示所属从站设备的ID号
* unsigned short m\_usChannelNum 表示该子模块下的通道个数。
* unsigned short m\_usStartChannelNO表示该子模块的起始通道号
* 类CChannel：从站设备下的通道信息。

主要成员：

* unsigned short m\_ushChannelNum表示通道号
* CString m\_strAddress 表示通道的直接地址
* emDataType m\_emChannelType表示通道的数据类型
* CString m\_strPointType表示通道的测点类型

1. 显示数据结构

硬件配置的框架采用Doc/View的结构设计，如下：



图 5：硬件配置框架设计

显示数据包括视图和对话框：

1. 视图类

1）CHardWareView：机柜、槽位组态视图。数据来自容器CHWFrameContainer，如下：



图 6：CHardWareView视图数据来源

* 类CHWFrameContainer：显示数据的容器

主要成员：

* CList<UINT,UINT> m\_shelfIDList表示组态的机柜对象ID列表
* CMap<UINT,UINT,CGraph\*,CGraph\*> m\_elementMap表示组态的图形元素
* 类CGraph：图形基类

主要成员：UINT m\_uiID 表示图形元素的ID

* 类CShelf：机柜信息

主要成员：

* CString m\_strName 表示机柜名字
* CList<UINT,UINT>\* m\_pFaceIDList表示包含的面ID列表
* 类CSlot：槽位信息

主要成员：

* UINT m\_uiParent表示所在机柜的ID
* CString m\_strText表示槽位显示的文本信息
* UINT m\_uiGroupID表示所在组的ID
* UINT m\_uiLogicID表示关联的逻辑数据ID
* 类CFace：机柜中的面信息，用于绘制机柜

主要成员：

* UINT m\_uiCabinetID表示所在机柜ID
* CList<UINT,UINT>\* m\_pGroupIDList表示包含的组ID列表
* 类CGroup：机柜中的组信息，用于绘制机柜

主要成员：

* UINT m\_uiCabinetID表示所在机柜ID
* UINT m\_uiFaceID表示所在的面ID
* CList<UINT,UINT>\* m\_pUnitIDList表示包含的单元ID列表



图 7：图形元素继承关系

2）CHWDPPALinkView：单个设备属性视图类。数据来自源数据及逻辑数据，如下：



图 8：CHWDPPALinkView视图数据来源

1. 对话框

对话框的数据来源根据其功能决定其来自于源数据、显示数据以及逻辑数据容器。这里不再详述。

模块内部的数据集中在CHWFrameContainer、CHWDataContainer和CHWSourceCointaner三个对象中。前两个对象构成主容器CHWContainer（每个控制站对应一个CHWContainer对象指针）的主要数据成员，而CHWContainer类派生于CBaseDPContainer类，CBaseDPContainer类中包含硬件配置与其他模块交互的接口及硬件配置内部三个DLL之间的交互接口，其对象存放在公共部分的CStation类中。



图 9：CHWContainer容器

1. 外部数据结构设计

硬件配置主要与变量、框架模块间进行数据交互，交互过程中使用到的重要数据结构有：

1. 增删改、打开模块时：传递硬件组态的详细信息，存放在公共部分的CStation类中。

typedef struct tagHWChannelAddrInfo

{

USHORT usChannelNO; //当前的通道号

BYTE byChInOutType; //通道输入输出类型，输入为DR\_INPUT，输出为DR\_OUTPUT

UCHAR ucChannelBitCount; //通道的位大小

UINT uiChannelBitOffset; //通道地址的位偏移

}HWChannelAddrInfo;

typedef struct tagHWModuleInfoForVar

{

char cModuleName[70]; //子模块名

BOOL bFixedFlag; //是否定位

BYTE byModuleActionType; //对子模块的动作类型：删除、修改

USHORT usModuleChannelNum; //子模块的通道个数

USHORT usOldStartChannelNo; //修改前子模块起始通道号

USHORT usNewStartChannelNo; //修改后子模块起始通道号

UCHAR ucInputBitOffset; //子模块的输入区位偏移

UCHAR ucOutputBitOffset; //子模块的输出区位偏移

UCHAR ucInputBitCount; //子模块输入区位大小

UCHAR ucOutputBitCount; //子模块输出区的位大小

CMap<USHORT, USHORT, CString, LPCTSTR>\* pPointTypeForChannelMap; //各通道对应的测点类型，关键字为通道号

CMap<USHORT, USHORT, emDataType, emDataType>\* pDataTypeForChannelMap; //各通道对应的数据类型，关键字为通道号

CList<HWChannelAddrInfo\*, HWChannelAddrInfo\*&>\* pChannelAddrInfoList; //各通道的位偏移信息

}HWModuleInfoForVar;

typedef struct tagHWHWPerSlaveInfoForVar

{

char cSlaveName[32]; //从站名

char cRedu; //从站是否冗余

BYTE byOldSlaveAddr; //修改前的从站站号

BYTE byNewSlaveAddr; //修改后的从站站号

BOOL bIsSlaveConfig; //从站是否可配置

USHORT usChannelNum; //从站的通道个数

CList<HWModuleInfoForVar\*, HWModuleInfoForVar\*&>\* pModuleForSlaveList; //此从站内的子模块列表

}HWPerSlaveInfoForVar;

typedef struct tagHWSlaveInfoForVar

{

USHORT usSlaveNum; //从站个数

emActionType byActionType; //动作类型：添加模块、删除模块、修改从站地址、查看测点、配置子模块

HWPerSlaveInfoForVar\* pHWPerSlaveInfoForVar; //各从站的信息

} HWSlaveInfoForVar;

1. 显示、修改通道参数信息时：传递设备的通道参数信息，存放在公共部分的CStation类中。

typedef struct tagHWCHParamInfoForPerCh

{

UINT uiValueID; //通道参数对应的索引

char chEditStatus; //参数编辑的状态：-1：不使能；0：使能

UINT uiValue; //当前参数值：类型为Combox时表示参数值的索引，为Edit时为参数值

CMap<UINT,UINT,CString,CString>\* pCHValueMap; //Combox：可选值的名称,关键字为可选值的索引，Edit：可选值

}HWCHParamInfoForCNPerCh;

typedef struct tagHWPerCHParamInfo

{

char chParamName[48]; //参数名称

char chEditType; //参数值编辑的类型：1: Combox ; 0:Edit

CMap<UINT,UINT,HWCHParamInfoForCNPerCh\*,HWCHParamInfoForCNPerCh\*>\* pRefValueForCHNOMap; //每行通道与参数信息的对应关系,关键字为通道号

}HWPerCHParamInfo;

typedef struct tagHWCHParamInfoForVar

{

char chSlaveName[32]; //从站名

BYTE bySlaveNO; //从站站号

UINT uiCHShowNum; //显示的通道参数个数

HWPerCHParamInfo\* pHWPerCHParamInfo;//每条通道参数信息

}HWCHParamInfoForVar;

1. 保存工程时：传递给总控当前控制站组态的硬件设备信息，存放在容器CBaseDPContainer中。

typedef struct tagSlaveConfigInfo

{

DWORD dwType; //数据类别

DWORD dwLength; //下文数据的长度

char\* pBuf; //数据内容

}SlaveConfigInfo;

typedef struct tagAddInfoPerSlave

{

BYTE byAddress; //模块站号

char cName[32]; //模块名

char cIsRedun; //是否冗余

BYTE byTextNum; //字符串信息的个数

SlaveConfigInfo\* pTextInfo; //字符串内容

}AddInfoPerSlave;