硬件配置

# 硬件配置模块

## 源数据DLL

源数据DLL：是硬件配置的源泉，用来实现配置文件、GSD文件、初始化文件的分析等，以供逻辑数据DLL和显示数据DLL组态使用

## 逻辑数据DLL

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。

## 显示数据DLL

显示数据DLL：界面显示

## 三者关系



图1

# 源数据

## 源数据组成

源数据的组成如下图所示。

图2：源数据的组成

数据源的信息依赖于三类磁盘文件：GSD文件、CFG文件及INI文件。

* 1. GSD文件是硬件模块的配置文件，该文件定义了模块的用户参数的默认值，以及模块的配置数据和通道信息等内容。GSD文件将会有两次读取的过程，第一次在程序启动时读取，第二次在硬件配置组态，或者导入GSD文件，或者序列化读取时进行读取
  2. CFG文件规定了机笼的配置信息、模块的信息以及控制器的链路配置等，并规定了机笼、控制器、模块等的图片名称和设备描述信息、设备库的显示信息等内容。CFG文件在程序启动的时候读取。
  3. INI文件分为两类：

ModuleChannel.ini文件规定了模块需添加的module的个数及在GSD文件中的序号、模块中通道的个数及每个通道的数据类型、输入输出类型和相对偏移、模块中每个通道对应的测点类型。启动AT时读取。

ChannelParam.ini文件规定了模块中每个通道对应的模块参数在所有参数字节中的相对偏移。在读取CFG文件之后读取。ITCC无此文件

以上数据都存放在类CHWSourceContainer中，类CHWSourceContainer的对象g\_SourceContainer作为全局对象存在。



图3：CHWSourceContainer类的主要成员

## GSD文件处理

### CGSD文件格式



### CGSD类

CGSD类对应着GSD文件库的内容。



图4：CGSD类的主要成员

### 第一次读取SearchDirectory

**入口函数**：void CGsd:: SearchDirectory(CString strDirectory)

**功能简述：**搜索特定目录下的所有GSD文件，分析文件信息，将分析结果记录到设备列表m\_gsdBaseMap；另外，需要把设备名和GSD文件名信息填入到CHWDesc::m\_itemMap中。

**参数说明：**CString strDirectory \*.gsd文件所在的目录"HardWare\\PCBasedIO\\"

**返回值：**无

**被调用：**AT启动时显示数据DLL显示树形控件CLibTreeCtrl时调用

**数据结构：**

* m\_gsdBaseMap：GSD对应设备的基本信息，用于判断设备是否主站等，全局

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

typedef struct tagDP\_BASE\_GSD

{

CString strDeviceName; //GSD文件中的设备名称

char chType; //设备的类型，0 从站，1 主站

}DP\_BASE\_GSD;

* m\_ModInfoMap：从ModuleChannel.ini中读出来的各模块信息，键值为模块名

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

图5：SearchDirectory流程图

### 第二次读取GetGsdFilePara

**入口函数：**char CGsd::GetGsdFilePara(CString FileName)

**函数功能：**根据GSD文件名称，分析GSD文件中描述的内容，并将分析到的设备特性记录到GSD的参数列表m\_gsdMap中

**参数说明：**CString FileName GSD文件名

**返回值：**如果分析正确，返回1（GSDFILE\_OK）；如果gsd数据已经存在，返回0（GSDFILE\_EXIST）；如果gsd数据错误，返回2（GSDFILE\_ERROR）

**被调用：**导入GSD文件时框架调用显示数据OnImportGsd，OnImportGsd调用该函数

**数据结构：**

* m\_gsdMap ：GSD文件对应的内存数据，全局

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

图6：m\_gsdMap结构

* m\_SlaveGSDInfoMap ：GSD文件中各模块的module信息，全局

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

typedef struct tagSlaveGSDInfo

{

Char cSlaveName[33]; //模块名

unsigned short usModuleNum; //module数

ModuleCfgInfo\* pModuleInfo; //module信息数组

}SlaveGSDInfo;

typedef struct tagCHInfoForModule

{

UCHAR uchInOutType; //输入输出类型，为输入型、输出型、输出/输入型

emDataType emINCHType; //输入通道数据类型

emDataType emOUTCHType; //输出通道数据类型

UINT uiINDataLenth; //输入数据长度

UINT uiOUTDataLenth; //输出数据长度

}CHInfoForModule;



图7：GetGsdFilePara流程图

## ini文件处理

### ChannelSOEPrms.ini

#### 文件格式

路径：HOLLIAS\_MACS\ITCC\_Auto Think\Target\HardWare\ ChannelSOEPrms.ini

[General]

Count=8 \\模块个数

[Mod1] \\模块1

Name=SGM410 \\模块名称

ChEnableNum=16 ; \\通道使能参数个数

ChSOEEnableNum=0 ;\\通道SOE使能参数个数

ChDiagEnableNum=16 ;\\通道连接诊断使能参数个数

Ch1=2.0 ; \\通道对应的第一个module中的参数字节索引(从0开始)，有三种形式：1.单个数字（1个字节），代表字节偏移；2.0.0（1位），代表第0个字节的第0位；3.0:2（多个字节），代表从第0个字节开始的2个字节

Ch2=2.1 [\\2](file:///\\2)对应GSD文件中Ext\_User\_Prm\_Data\_Ref(2) = 13的索引2

Ch3=2.2

Ch4=2.3

Ch5=2.4

Ch6=2.5

Ch7=2.6

Ch8=2.7

Ch9=3.0

Ch10=3.1

Ch11=3.2

Ch12=3.3

Ch13=3.4

Ch14=3.5

Ch15=3.6

Ch16=3.7

ChDiag1=4.0

ChDiag2=4.1

ChDiag3=4.2

ChDiag4=4.3

ChDiag5=4.4

ChDiag6=4.5

ChDiag7=4.6

ChDiag8=4.7

ChDiag9=5.0

ChDiag10=5.1

ChDiag11=5.2

ChDiag12=5.3

ChDiag13=5.4

ChDiag14=5.5

ChDiag15=5.6

ChDiag16=5.7

#### 读取ReadChSOEPrmsInfo

**入口函数**：void CHWSourceContainer::ReadChSOEPrmsInfo()

**功能简述：**从ChannelSOEPrms.ini解析通道使能、通道SOE使能、连接诊断使能信息

**返回值：**无

**被调用：**AT启动时构造CHWSourceContainer时调用

**数据结构：**

* m\_ChSOEPrmsInfoMap：各module通道与module参数对应关系

CMap<CString,LPCTSTR,ChSOEPrmsInfo\*,ChSOEPrmsInfo\*&> m\_ChSOEPrmsInfoMap

typedef struct tagChSOEPrmsInfo

{

char cName[33]; //模块名

unsigned short usChEnableNum; //通道使能参数个数

unsigned short usChSOEEnableNum;//通道SOE使能参数个数

unsigned short usChDiagEnableNum;//通道连接诊断使能参数个数

ParamOffset\* pEnablePrmIndex; //每个通道对应的使能参数在module所有参数中的索引(从0开始)

ParamOffset\* pSOEEnablePrmIndex;//每个通道对应的SOE使能参数在module所有参数中的索引(从0开始)

ParamOffset\* pDiagEnablePrmIndex;//每个通道对应的连接诊断使能参数在module所有参数中的索引(从0开始)

}ChSOEPrmsInfo;

typedef struct tagParamOffset

{//通道对应的参数的偏移，eg: Ch1=2.0

unsigned char ucByteCount;//参数的字节数，多字节为字节总个数，单字节为1，位为0

unsigned char ucByteOffset;//参数的字节偏移 eg：2

unsigned char ucBitOffset;//参数的位偏移 eg：0

}ParamOffset;



图8：ReadChSOEPrmsInfo流程图

### HardWareSetting.ini

#### 文件格式

路径：HOLLIAS\_MACS\ITCC\_Auto Think\Target\HardWare\ HardWareSetting.ini

[General]

IsDPImportSupported = false \\是否可以导入GSD文件

IsCageAddr80Supported = false \\是否可以将扩展机柜的拨码改到80及以上

IsCageAddrModifySupported = false\\是否可以修改扩展机笼的拨码

#### 读取ReadHWSettingInfo

**入口函数**：void CHWSourceContainer::ReadHWSettingInfo()

**功能简述：**从HardWareSetting.ini中解析出是否可以导入DP设备文件、机柜拨码是否可以设置为80及以上、是否可以修改扩展机笼的拨码

**返回值：**无

**被调用：**AT启动时构造CHWSourceContainer时调用

**数据结构：**

* m\_HWSetting：从HardWareSetting.ini中读取的信息

HardWareSetting m\_HWSetting;

typedef struct tagHardWareSetting

{

bool bIsDPImportSupported; //是否可以导入GSD文件

bool bIsCageAddr80Supported;//扩展机笼的拨码是否可以改到80及以上

bool bIsCageAddrModifySupported;//是否可以修改扩展机笼的拨码

}HardWareSetting;



图8：ReadHWSettingInfo流程图

### ModuleChannel.ini

#### 文件格式

[General]

Count=9 \\模块总个数

[Mod1] \\模块1

Name=SGM410 \\模块名称

ChannelNum=16 \\通道个数

ChildModNum=2 \\子模块个数

ChildMod1=1 [\\子模块1](file:///\\子模块1)在GSD文件索引1

ChildMod2=2 \\子模块2在GSD文件索引2

Ch1=AI3CH,1,23,2,0 \\通道信息：1.数据库点类型; 2.数据类型(0:BIT; 1:WORD; 2:REAL; 3:LONG); 3.输入还是输出(23:输入; 5:输出); 4.字节偏移; 5.位偏移。

Ch2=AI3CH,1,23,4,0

Ch3=AI3CH,1,23,6,0

Ch4=AI3CH,1,23,8,0

Ch5=AI3CH,1,23,10,0

Ch6=AI3CH,1,23,12,0

Ch7=AI3CH,1,23,14,0

Ch8=AI3CH,1,23,16,0

Ch9=AI3CH,1,23,18,0

Ch10=AI3CH,1,23,20,0

Ch11=AI3CH,1,23,22,0

Ch12=AI3CH,1,23,24,0

Ch13=AI3CH,1,23,26,0

Ch14=AI3CH,1,23,28,0

Ch15=AI3CH,1,23,30,0

Ch16=AI3CH,1,23,32,0

#### 读取ReadModChannelInfoToMap

**入口函数**：void CHWSourceContainer::ReadModChannelInfoToMap()

**功能简述：**从ModuleChannel.ini解析模块通道信息,并将分析到的设备特性记录到GSD的参数列表m\_ModInfoMap中

**返回值：**无

**被调用：**AT启动时构造CHWSourceContainer时调用

**数据结构：**

* m\_ModInfoMap：从ModuleChannel.ini中读出来的各模块信息，键值为模块名

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

typedef struct tagModInfo

{

char cModName[33]; //模块名

unsigned char ucChildModNum; //要添加的子模块个数

unsigned char\* pChildModNo; //各子模块对应的在GSD文件中的索引值

unsigned short usChNum; //通道个数

ChInfo\* pChInfo; //各通道信息

}ModInfo;

typedef struct tagChInfo

{

char cPointType[33]; //通道对应的测点类型

int iChType; //通道类型，0:BIT; 1:WORD; 2:REAL; 3:LONG

int iRefID; //通道所在区域，1:输入; 2:输出

int iByteOffset; //通道对应的字节偏移

int iBitOffset; //通道对应的位偏移

}ChInfo;



图9：ReadModChannelInfoToMap流程图

### ModulesCommPrms.ini

#### 文件格式

[Module.SGM216]

Name=SGM216

Param1=Name=Master\_Slave,Section=MasterOrSlave

Param2=Name=Slave\_Address,Section=ModbusSlaveAddress

Param3=Name=Baud\_Rate,Section=BaudRate\_Type

Param4=Name=Parity,Section=VerifyMode\_Type

Param5=Name=Stop\_Quantity,Section=StopBit\_Type

Param6=Name=Polling\_Interval,Section=PollingInterval\_Type

Param7=Name=Timeout\_Period,Section=Timeout\_Period\_Type

Param8=Name=Resend\_Times,Section=Resend\_Times\_Type

;Param5=Name=DataLength,Section=DataLength\_Type \\去掉数据长度这个参数，硬件写死为RTU协议，即8位

[SubModule.1]

Name=读线圈(0xxxx,01H)

Param1=Name=Modbus Slave Address,Section=ModbusSlaveAddress \\从站号

Param2=Type=BYTE,Value=0x01 \\功能码，固定值

Param3=Name=Modbus Start Address,Section=ModbusStartAddress \\从站起始地址

Param4=Name=Modbus Read/Write Bits Length,Section=R\_W\_Bit\_Length \\读写开关量长度

Param5=Type=BYTE,Value=0 \\是否变位传输，固定值。0:不变位,1:变位

[SubModule.2]

Name=读离散量输入(1xxxx,02H)

Param1=Name=Modbus Slave Address,Section=ModbusSlaveAddress

Param2=Type=BYTE,Value=0x02

Param3=Name=Modbus Start Address,Section=ModbusStartAddress

Param4=Name=Modbus Read/Write Bits Length,Section=R\_W\_Bit\_Length

Param5=Type=BYTE,Value=0

……

[Parameter.Variable\_Bit]

Id=10012

Type=BYTE

Class=Symbolic

Symbol1=0

Value1=0

Symbol2=1

Value2=1

Default=1

……

[Parameter.ModbusStartAddress]

Id=10009

Type=WORD

Class=Simple

Min=0

Max=65535

Default=0

[Parameter.MasterOrSlave]

Id=10005

Type=BYTE

Class=Symbolic

Symbol1=Modbus Master

Value1=1

Symbol2=Modbus Slave

Value2=0

Default=2

[Parameter.ModbusSlaveAddress]

Id=10004

Type=BYTE

Class=Simple

Min=1

Max=247

Default=1

[Parameter.BaudRate\_Type]

Id=10000

Type=BYTE

Class=Symbolic

Symbol1=115.2Kbps

Value1=1

Symbol2=57.6Kbps

Value2=2

Symbol3=38.4Kbps

Value3=3

Symbol4=19.2Kbps

Value4=4

Symbol5=9600bps

Value5=5

Symbol6=4800bps

Value6=6

Symbol7=2400bps

Value7=7

Symbol8=1200bps

Value8=8

Default=5

[Parameter.VerifyMode\_Type]

Id=10003

Type=BYTE

Class=Symbolic

Symbol1=none

Value1=1

Symbol2=odd

Value2=2

Symbol3=even

Value3=3

Default=3

[Parameter.StopBit\_Type]

Id=10001

Type=BYTE

Class=Symbolic

Symbol1=1 stop bit

Value1=1

Symbol2=2 stop bit

Value2=2

Default=1

[Parameter.PollingInterval\_Type]

Id=10006

Type=BYTE

Class=Symbolic

Symbol1=50ms

Value1=1

Symbol2=200ms

Value2=2

Symbol3=500ms

Value3=3

Symbol4=1000ms

Value4=4

Symbol5=2000ms

Value5=5

Default=1

[Parameter.Timeout\_Period\_Type]

Id=10007

Type=BYTE

Class=Symbolic

Symbol1=50ms

Value1=1

Symbol2=200ms

Value2=2

Symbol3=500ms

Value3=3

Symbol4=1000ms

Value4=4

Symbol5=2000ms

Value5=5

Symbol6=3000ms

Value6=6

Default=3

[Parameter.Resend\_Times\_Type]

Id=10008

Type=BYTE

Class=Simple

Min=0

Max=10

Default=2

[Parameter.DataLength\_Type]

Id=10002

Type=BYTE

Class=Symbolic

Symbol1=7 bit

Value1=0

Symbol2=8 bit

Value2=1

Default=2

#### 读取GetAllComModPrmInfo

**入口函数**：char CComModPrmInfo::GetAllComModPrmInfo()

**功能简述：**查找ModulesCommPrms.ini文件对应的所有通讯模块的全部参数信息并读取到m\_ModComPrmInfoMap中

**返回值：**读取成功返回1，否则返回0

**被调用：**AT启动时构造CHWSourceContainer时调用

**数据结构：**

* m\_ModComPrmInfoMap：从ModulesCommPrms.ini中读出来的各模块信息，键值为模块名

CMap<CString,LPCTSTR,MOD\_COM\_PRMINFO\*,MOD\_COM\_PRMINFO\*&> m\_ModComPrmInfoMap;



## CFG文件处理

### CFG文件格式

