# 交叉引用表

交叉引用表（CCrossTable）是设置POU编译和变化标志的依据，当某个变量发生变化时就要通过该表索引到引用该变量的所有POU，设置POU编译和变化的标志；

## 交叉引用表数据结构如下



有了上述的交叉引用表的结构图，下面对交叉引用表的操作做简单介绍：

## 删除POU名字对应的交叉引用项(IEC调用)：

void DeletePOUCrossTable(CString strPOUName)



## 添加POU对应的交叉引用项(IEC调用):

char AddPOUCrossTable(CString strPOUName,CrossRecordList\* pList)



## 更新交叉引用表

m\_crossPOUTableMap、m\_crossVarTableMap: void UpdataCrossInfoByPOU()

主要完成两项任务

* 由pou名字对应的交叉引用表生成变量对应的交叉引用表
* 检查有没有删除的POU，如果有就要相应的删除此POU对应的交叉引用信息；

注：此过程中的“清除变量名对应的MAP中的交叉引用信息”没有必要，因为UpdataCrossInfoByPOU()是在序列化的时候执行的，在序列化之前已经将所有的交叉引用表进行清空了，当然也包括清楚变量名对应的交叉引用表信息，如果该序列化之前没有调用DelAllRecord()，在此调用RemoveVarMap()也是没有用处的，因为RemoveVarMap()函数中只是删除了链表的信息，至于链表里面的指针指向的信息没有删除。

删除变量索引表: void RemoveVarMap()

因为变量和POU公用对应的交叉引用记录，所以只需要在一处删除即可；

## 删除所有交叉信息: void DelAllRecord()

调用情况：

* + 析构函数
  + 反序列化之前

此功能是综合了删除POU名字对应的交叉引用项和删除变量索引表；

## 根据交叉引用信息设置POU标志:

bool SetPOUFlag(CString strVarName)

调用情况：

* 增加和删除自定义数据类型；
* 删除、重命名PRG类型的POU时删除和重命名同名的全局变量组时；
* 删除某个已经定义的变量组；



## 设置多个变量发生变化时的POU编译和修改标志

bool CCrossTable::SetPOUFlagArray(CStringArray\* pStrVarNameArray)

调用情况

* + 导入全局变量
  + 导入测点变量时