福建西门子代理商SIEMENS

产品名称	福建西门子代理商SIEMENS
公司名称	上海领国自动化科技有限公司
价格	.00/件
规格参数	型号:SIEMENS/plc模块 中国:代理商 售后:技术支持
公司地址	上海市金山区枫泾镇环东一路65弄7号3959室
联系电话	18800378001

产品详情

福建西门子代理商SIEMENS

其中第0～2位（xxx，*低位为第0位）为被寻址地址中位的编号（0～7），第3～18位（16个b）为被寻址地址的字节的编号。第24～26位（rrr）为被寻址地址的区域标识号，指针的*高位x为0时，为区域内的间接寻址，*高位x 为1时，为区域间（交叉区域）间接寻址。 参数类型ANY可以用来传递一片连续的地址区，由10个字节组成。ANY和POINTER用于在块调用时传递输入、输出参数。为了揭开V区之谜，编写了FC1，将地址区中相邻的若干个字累加。地址区的起始地址由参数类型为POINTER的输入参数Start_Addr提供。P#DB2.DBX0.0也可以改写为DB2.DBX0.0。在OB1中调用FC1： CALL FC 1 Start_Addr :=P#DB2.DBX0.0 //数据区起始地址 Number :=5 //需要累加的字数 Result :=DB2.DBD10 //保存运算结果的双整数 图1是运行时监控FC1的结果，累加器1（STANDARD）中的数据为十六进制显示格式，AR1是地址寄存器1。终于看到了AR1中的V区地址了！

图1中第一条指令的P#表示指针，第2个#号表示局部变量。P##Start_Addr就是调用FC1时，用输入参数Start_Addr传送给FC1的指针P#DB2.DBX0.0（16#0002 8400 0000）存放的地址。P##Start_Addr（16#8700 00a8）*低字节16#a8对应的二进制数为2#10101000，其字节部分为2#10101，即十进制数21，*高字节16#87（2#1000 0111）表示存储区为V区。 第一条指令将P##Start_Addr送给累加器1，第二条指令将累加器1中的数据传送到AR1，传送后AR1中的地址为V21.0（即16#8700 00a8）。
那么V区到底是什么呢？根据帮助中的解释“ 先前的本地数据 ”（Previous local data），猜想与局部数据堆栈有关。执行每个块时，它都有自己的临时局部数据。在OB1调用FC1时，OB1的临时局部数据被保存到局部数据堆栈，FC1则使用它自己的临时局部数据区，OB1的局部数据成为“ Previous local data ”（以前的局部变量）。根据上述分析，V区很有可能是调用FC1的OB1的局部数据区。
怎样才能证实这个猜想呢？**能看到AR1中的地址为V21.0时，OB1的局部数据。好在STEP 7的监控功能可以查看块调用时保存在堆栈中的数据。为了能看到某条指令执行后OB1的局部数据，在FC1的第2条指令处设置一个断点。执行完第2条指令后，CPU进入HOLD模式，此时打开CPU模块信息对话框的“ 堆栈

”选项卡，选中B堆栈中的OB1，点击“L堆栈”按钮，打开L堆栈对话框，OB1的局部数据堆栈如图2所示。

由图1可知，因为指针常数P#V21.0（16#8700 00a8）被送给AR1，监控区中的AR1列显示V21.0。此时OB1调用FC1的POINTER格式的实参P#DB2.DBX0.0（16#0002 8400 0000），存放在从OB1的局部变量LB21开始的6个字节中（见图2）。因此AR1中的P#V21.0表示指针常数P#DB2.DBX0.0的值存放在OB1的局部变量区中的地址，换句话说，V区就是调用FC1时OB1的局部数据区。
难怪“没见有谁用这个区域编程”，V区用于监控，在编程时没有使用它。 *后我们来总结一下块调用时的参数传递过程。如果输入参数为简单数据类型，例如字节、字、整数和双整数，可以通过32位（4个字节）的累加器1直接传递参数。而ANY和POINTER分别为10个和6个字节，不能用累加器1直接传递。因此将这些参数的实参（例如16#0002 8400
0000）暂时保存在OB1从V21.0开始的局部变量中。在被调用的FC1中，
P##Start_Addr提供了保存参数Start_Addr的实参的地址V21.0，在FC1中用寄存器间接寻址指令“L W [AR1,P#0.0]”来读取POINTER实参的第一个字（数据块编号），用指令“L D [AR1,P#2.0]”来读取POINTER实参的2~5号字节（数据块内的变量地址P#
DBX0.0）。间接寻址的操作数地址等于方括号中AR1的地址值加上逗号后面的地址偏移量。
说到这里，我们可以看到传递POINTER参数类型的思路是非常清晰的，“Previous local data”用词是准确的，只不过所用的笔墨太少，背后的复杂过程需要我们猜想和验证。
解决了这个问题后，有一些感触： 1．由于语言和思维方式的差异，老外写的用户手册有的地方很难理解，这并不奇怪。奇怪的是网上有一些高手的“用户手册**论”。用户手册肯定不是**的，不可能回答所有的问题，有的问题还需要我们设法去探索和发现，包括用程序来验证我们的假设。
2．这个问题的解决使我惊叹STEP 7强大的功能，如果没有断点和监控堆栈的功能