SIEMENS

西门子 STL 间接寻址常问问题集

Cluster-FAQ for Indirect Addressing in STL

Cluster-FAQ	**Edition (2011 年 12 月)**

摘　要 间接寻址可以实现程序执行时动态修改程序变量的地址和内容, 从而有效地提升程序的灵活性和适用范围, 但是由于地址值只会在运行时才可以确认, 因此错误的间接寻址会导致错误的结果甚至导致程序的异常响应, 工程师需要正确的使用间接寻址, 必须避免一些常见的问题, 本文使用问答的方式为学习和使用间接寻址功能的工程师参考

关键词 间接寻址，指针，地址寄存器

Key Words indirect addressing，pointer，address register

目　录

1.1 如何获得指针或者间接寻址有关的信息?

指针的类型包括 16 位指针、32 位指针、Pointer（6Byte）和 Any（10Byte）。16 位指针用于定时器、计数器、程序块的寻址；32 位指针用于 I/Q/M/L/数据块等存储器中位、字节、字以及双字的寻址，其中第 0~2 位表示位地址（0~7）、第 3~18 位为字节地址，其余位未定义；Pointer 和 Any 一般应用在复杂数据类型（比如 Date_and_Time /Array/String 等）在 FB、FC 之间的传递。而 Any 可以看做是对 Pointer 的延伸，因为由 10Byte 组成的 Any 中 Byte4~Byte9 就是一个 Pointer。

了解指针的格式十分重要，为正确使用指针，应阅读如下内容：

1、 "SIMATIC Programming with STEP 7 V5.5" 05/2010 第 27.3.4 章 参数类型

2、文档：1008 用于 S7-300 和 S7-400 的语句表(STL)编程

3、文档：F0215，S7-300 和 S7-400 寻址

1.2 为什么语句 LAR1 P##PointerInput 在一个函数(FC)中是无效的,然而，同样的语句在一个功能块(FB)中是有效的?

在 FC 被调用时，复杂数据类型例如指针是被复制到调用者的临时变量区中，在 FC 内部对此 V 区地址直接取址放入到地址寄存器 AR1 或 AR2 是不被编译器规则接受的（导致 MC7 寄存器信息过长），也就是说在 FC 内部通过 P#进行地址寄存器取址仅能支持 Temp 临时变量。因此如果需要在 FC 中操作指针等复杂输入输出变量地址需要使用累加器进行中转。

考虑到程序的一致性、遵守编译器规则和 STL 手册中 LAR1 指令说明，建议用户使用如下指令操作：

```
L P##PointerInput
LAR1
```

1.3 STEP 7 中哪些操作会覆盖 DB/DI 寄存器或者地址寄存器 AR1/AR2 的内容?

下面说明了可能引起 DB/DI 寄存器或者地址寄存器 AR1/AR2 内容改变的一些操作：

- DB 寄存器和 AR1 受到影响的操作

  1. 使用完整的 DB 路径（如 L DB20.Val）或者调用 FC/FB 时使用 DB 块完整地址作为其参数，则 DB 寄存器内容被覆盖。

     例如在 OB1 中调用 FC1 后，DB 寄存器变成 20。

     ```
     OPN DB1
     ```

```
Call  FC1
Input(bit)：DB20.DBX0.2
```

因此在编程的时候，OPN 指令打开数据块，通过 DBX x.y 的方式访问其中内容，但是如果在打开数据块后 DB 寄存器的内容被修改了，则 DBX x.y 的方式访问变量则会访问到错误的地址。可以通过使用符号寻址的方式或者使用完整路径编程避免，当然重新使用 OPN 指令也是可以的。

2. 调用 FC 时使用 string，array，structure ，UDT 作为其形参或者调用 FB 时使用 string，array，structure 或者 UDT 作为其 in out 形参，在 FC/FB 程序中访问这些地址则 AR1 寄存器内容被覆盖，因此当使用 AR1 进行间接寻址时需要注意 AR1 内容的正确性。

- AR2 地址寄存器和 DI 寄存器在 FB 中作为参数和静态变量的基址寻址使用。AR2 和 DI 如果被修改，会影响 FB 的参数访问，如果希望在 FB 中使用 DI 寄存器或者地址寄存器 AR2，必须预先保存它们中的内容，并在使用后恢复它们，例如：

```
TAR2  #AR2_SAVE; //AR2 寄存器状态保存到#AR2_SAVE
 L     DINO;
T      #DB2_SAVE; //DI 寄存器状态保存到#DB2_SAVE

 User Program

LAR2  #AR2_SAVE; //AR2 寄存器恢复到使用前状态
OPN   DI [#DB2_SAVE]; //DI 寄存器恢复到使用前状态
```

1.4 如何得到多重背景 FB 中的变量在背景 DB 里的绝对偏移量呢？

可以用下面的方法处理：

```
TAR2                (得到多重背景 FB 在背景 DB 里的偏移地址)
AD DW#16#00FFFFFF   (屏蔽掉存储区 ID，可参考 32 位指针格式)
L P##Variable       (得到变量在多重背景 FB 里的地址)
+D (多重背景 FB 的偏移地址与变量在多重背景 FB 里地址相加，即得到实际绝对偏移量)
LAR1
```

上述语句就是就得到了变量在背景 DB 中的绝对偏移量，从而供后续程序处理。

1.5 如何在程序中使用 ANY 型指针？

简要说明如下：

```
   L      P##Input          //指向存储地址指针 Input 首地址
 //这个参数是一个 Any 类型，P##Input 指向参数 Input 的值所在地址，这就是指针的指针
   LAR1                     //装载到地址寄存器 AR1 中。
   L      W [AR1,P#4.0]     //打开 DB 块
//  由 Any 类型结构知道 Any 类型的 Byte4、Byte5 存放的数据块号
   T      #BLOCK_NO
   OPN    DB [#BLOCK_NO]    //如果是 DB 块，打开指定的 DB 块。
   L      W [AR1,P#2.0]     //判断 ANY 指针中数据长度
//  Any 类型的 Byte2、Byte3 是重复系数，如 P#DB1.DBX0.0 Byte 8 后面的 Byte 8
   _001:T      #DATA_LEN    //通常此处做 loop 循环！！
   L      D [AR1,P#6.0]     //找出需要计算数据区的开始地址
//  Any 类型 Byte6~Byte9 是 32 位区域地址
```

理解 Pointer、Any 的类型的数据结构，对于正确使用指针有很大帮助。

为正确使用指针，应仔细阅读如下内容：

"SIMATIC Programming with STEP 7 V5.5" 05/2010 第 27.3.4 章 参数类型

如下的程序实现了 SFC20 的部分功能，可以作为 Any 使用的参考。

```
FUNCTION FC 1 : VOID
 TITLE =
 VERSION : 0.1
  VAR_INPUT
    SRCBLK : ANY ;
 END_VAR
 VAR_OUTPUT
    RETVAL : INT ;
    DSTBLK : ANY ;
 END_VAR
 VAR_TEMP
    LOOP : INT ;
    BLOCK_NO_DB : WORD ;
```

```
  BLOCK_NO_DI : WORD ;
  SRC_ADD : DWORD ;
  DST_ADD : DWORD ;
END_VAR
BEGIN
NETWORK
TITLE =
       L     P##SRCBLK;    //读取输入 any 的首地址
      LAR1  ;                          //装载到 ar1
      L     P##DSTBLK;    //读取输出 any 的首地址
      LAR2  ;                          //装载到 ar2
       L     W [AR1,P#4.0];     //打开 DB 块
      T     #BLOCK_NO_DB;
      L     W [AR2,P#4.0];          //打开 DI 块
      T     #BLOCK_NO_DI;
      OPN   DB [#BLOCK_NO_DB];      //打开 DB 块
      OPN   DI [#BLOCK_NO_DI]; //打开 DI 块
       L     D [AR1,P#6.0];
      T     #SRC_ADD;    //读取地址
      L     D [AR2,P#6.0];
      T     #DST_ADD; //读取地址

      L     W [AR1,P#2.0]; //读取循环次数
_001: T     #LOOP;
       L     DBB [#SRC_ADD];
      T     DIB [#DST_ADD]; //赋值
 //地址偏移 1 个字节
      L     P#1.0;
      L     #SRC_ADD;
      +D    ;
      T     #SRC_ADD;
```

```
      L     P#1.0;
      L     #DST_ADD;
      +D    ;
      T     #DST_ADD;
      L     #LOOP;  //循环
      LOOP  _001;
END_FUNCTION
```

1.6 当 FC 或 FB 的输入参数类型为：BLOCK_DB，TIMER 或者 COUNTER，如何确定其编号？

例 1 ：FB 块

FB1 变量声明中定义了“ Timer ” 类型的变量“ Time_1 ” ，在 FB2 中调用 FB1，将定时器“ T5 ”传递给变量“ Time_1 ”。如图 01 所示程序代码中数值 5 表示“ T5 ”。

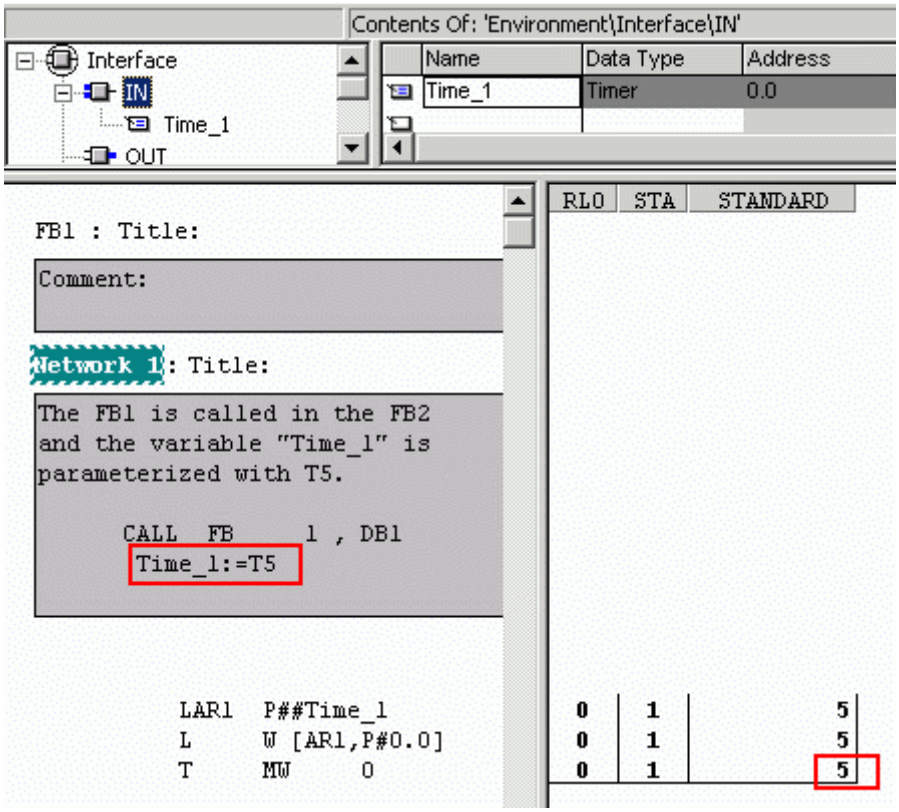

图 01 FB 中确定定时器编号

在使用多重实例时，需要在图 01 所示程序中增加以下代码：

```
TAR2                          //多重实例偏移地址
LAR1 P##Time_1
+AR1                          //多重实例偏移地址与当前地址相加
L W[AR1,P#0.0]
T MW0
```

例 2 FC

FC1 变量声明中定义了“ Timer ” 类型的变量“ Time_1 ” ，在 FC2 中调用 FC1，将定时器“ T8 ”传递给变量“ Time_1 ”。如图 02 所示程序代码中数值 8 表示“ T8 ”。

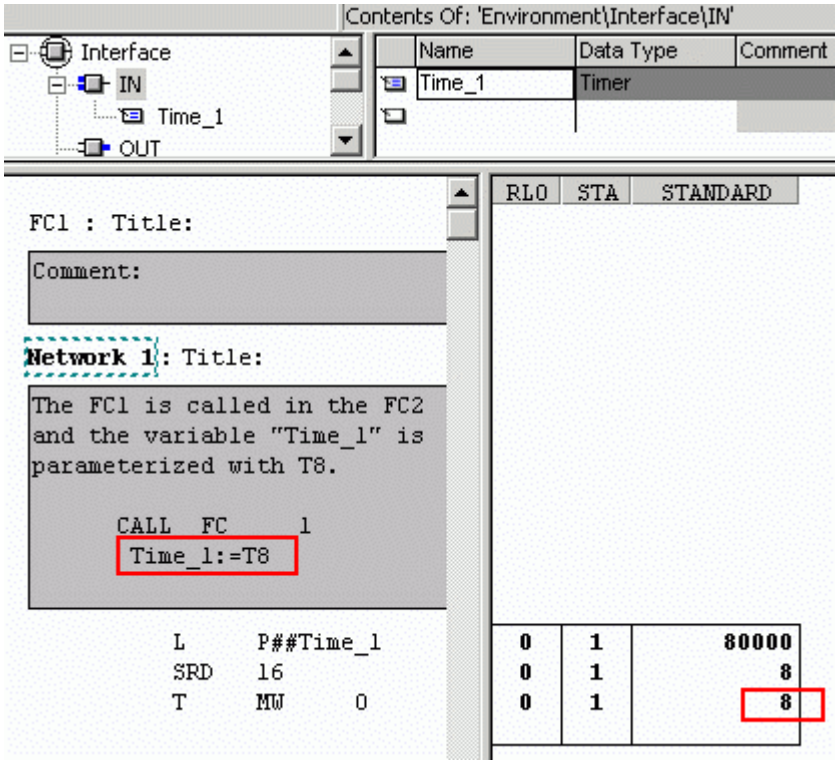

图 02 FC 中确定定时器编号

“ BLOCK_DB ”类型参数在调用 FC 时不可以直接传送给 FC 。如果在调用功能时试图使用此参数类型，将产生如下的错误消息： “ 非法的参数传输<参数名> ” 。“ BLOCK_DB ”类型参数仅在 FB 被调用（从一个 FB 或 FC 中）时可被传送。 “ BLOCK_DB ”类型参数不可以在 FC 被调用（从一个 FB 或 FC 中）时可被传送。

相关详细信息可参考 STEP7 在线帮助中“合法的传输参数类型”。

然而，如果希望将 BLOCK_DB 参数类型传送给 FC， DB 块的编号可以通过基本数据类型（例如 WORD）传送。 在下面的例子中，FB100 拥有一个“ BLOCK_DB ” 的输入参数类型。为了在调用 FC101 时将参数传送给它，“ BLOCK_DB ”中的 DB 块编号被传送给 WORD 临时变量(DB_No)。当 FC 被调用，数据块序号以一个 WORD 参数类型替代 BLOCK_DB 参数类型。

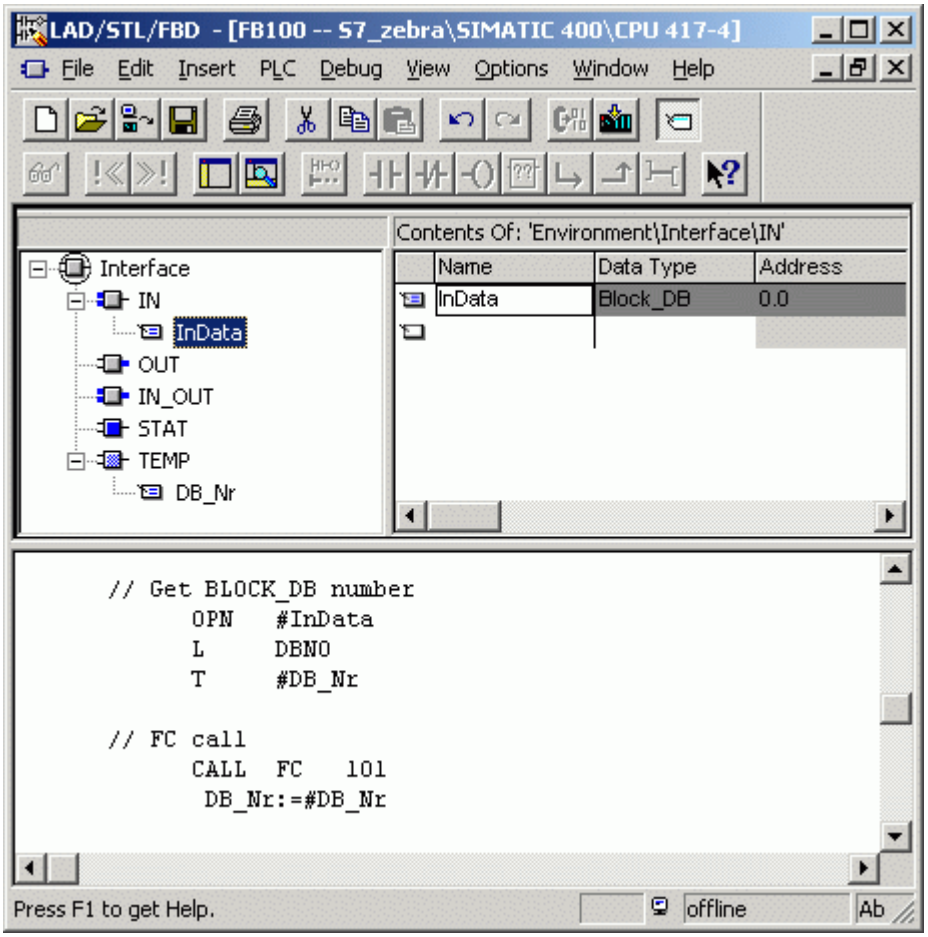

图 03 FB 中确定 Block_DB 编号

如果 FB 的接口参数能被定义成 WORD 参数类型来传送 DB 块编号，当 FC 被调用时，此参数可以被直接传送给 FC。

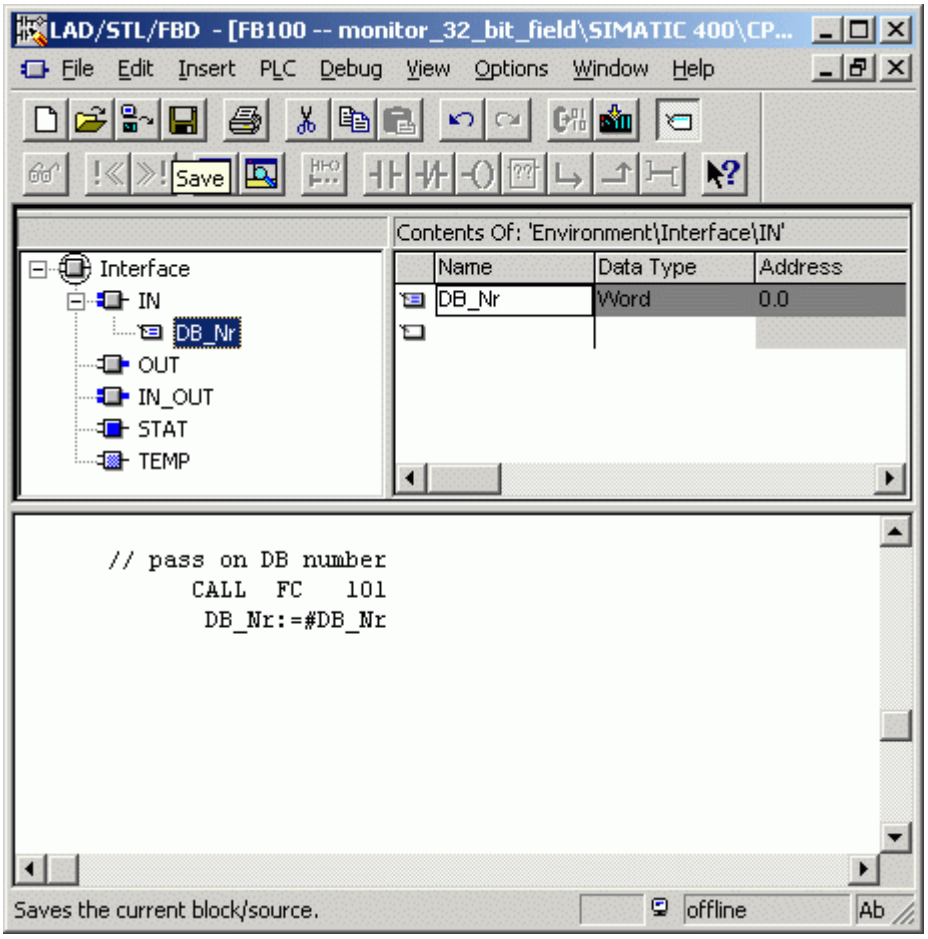

图 04 FC 中确定 DB 编号

1.7 参数传递有何限制?

当分配实际参数给形式参数时，可以指定绝对地址、符号名称或常数。STEP 7 限制不同参数的分配。例如，输出和输入/输出参数不能被分配常数值(因为输出或输入/输出参数的目的是改变其值)。这些限定尤其适用于具有复杂数据类型的参数，这些参数既不能分配绝对地址也不能分配常数。下表关明涉及分配给形式参数的实际参数数据类型的限制(--)和允许的分配(由● 符号显示)。

		基本数据类型		
声明类型	绝对地址	符号名称 (在符号表中)	临时本地符号	常数
输入	●	●	●	●
输出	●	●	●	--
输入/输出	●	●	●	--

		复杂数据类型		
声明类型	绝对 地址	DB 元素的符号名称 (在符号表中)	临时本地符号	常数
输入	--	●	●	--
输出	--	●	●	--
输入/输出	--	●	●	--

图 05 允许的参数传递

详细信息请参考手册中附录27.3.4.8节内容：
http://support.automation.siemens.com/CN/view/zh/45531107

1.8 如何传递 any 参数到其他程序块的参数中？

下面的例子将说明如何为系统功能 SFC50 “RD_LGADR” （读取模块逻辑地址）参数化 ANY 指针。例如对于功能块 FB1，按下述步骤编程：

1. 声明一个输入变量“test”和一个临时变量“test2”为 ANY 类型(图 05)。
2. 例如，把 SFC50 的参数“PEADDR”传递给变量“test2”(图 06)。
3. 通过判断 ANY 指针“test”,能够传递临时变量“test2”的数据。

FB1 -- SFC50_Test\SIMATIC 400(1)\CPU 414-2 DP

Address	Declaration	Name	Type	Initial val
0.0	in	test	ANY	
	out			
	in_out			
10.0	stat	LADDR_DECK	WORD	W#16#0
12.0	stat	RET_VAL_DECK	INT	0
14.0	stat	PEADDR_DECK	ARRAY[0..4]	
*2.0	stat		WORD	
24.0	stat	PECOUNT_DECK	INT	0
26.0	stat	PAADDR	ARRAY[0..4]	
*2.0	stat		WORD	
36.0	stat	PACOUNT_DECK	INT	0
0.0	temp	test2	ANY	

图 06 any 的临时变量传递

语句 L P##test 先把地址加载到 Accu1，然后通过语句 LAR1 把地址加载到地址寄存器 AR1（可简写为：LAR1 P##test）。每次读取地址寄存器 AR1 并存储数据（例如 T LW0）到临时变量“test2”（ANY 指针）中。Network 1 中的语句复制数据传送到功能块 FB1 的 ANY 数据到临时变量“test2”。

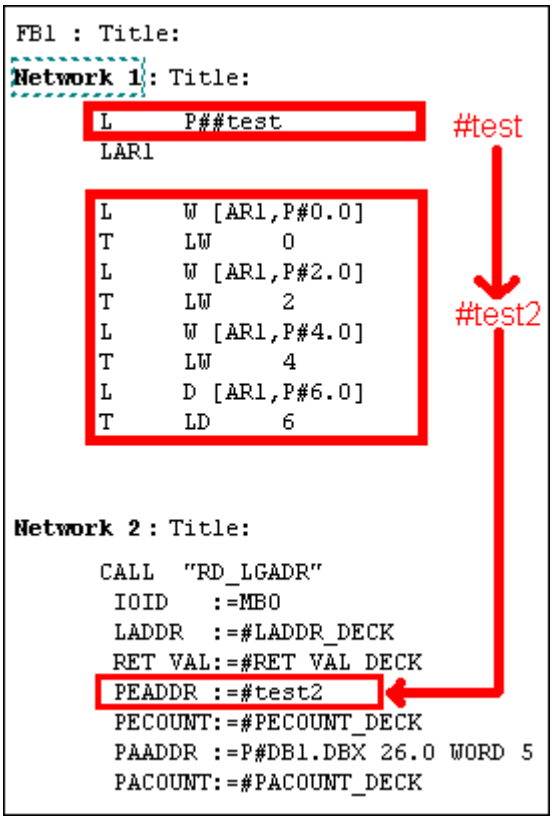

图 07 临时变量的建立

1.9 如何通过 UC 或 CC 指令调用 FB?

当使用 UC，CC 指令调用不带参数的 FB 可以通过手动修改 DI 值的方式进行背景数据块的动态分配，此操作相当于模仿系统调用 FB 的过程。

例如：
```
OPN DI[FB1_DI_Number]
LAR2 P#DBX0.0
UC FB1
```

如果您对该文档有任何建议，请将您的宝贵建议提交至下载中心留言板。

该文档的文档编号：F0595

附录一推荐网址

自动化系统
西门子（中国）有限公司
工业业务领域 客户服务与支持中心
网站首页：www.4008104288.com.cn
自动化系统 **下载中心**：
http://www.ad.siemens.com.cn/download/DocList.aspx?TypeId=0&CatFirst=1
自动化系统 **全球技术资源**：
http://support.automation.siemens.com/CN/view/zh/10805045/130000
"找答案"自动化系统版区：
http://www.ad.siemens.com.cn/service/answer/category.asp?cid=1027

注意事项

应用示例与所示电路、设备及任何可能结果没有必然联系，并不完全相关。应用示例不表示客户的具体解决方案。它们仅对典型应用提供支持。用户负责确保所述产品的正确使用。这些应用示例不能免除用户在确保安全、专业使用、安装、操作和维护设备方面的责任。当使用这些应用示例时，应意识到西门子不对在所述责任条款范围之外的任何损坏/索赔承担责任。我们保留随时修改这些应用示例的权利，恕不另行通知。如果这些应用示例与其它西门子出版物(例如，目录)给出的建议不同，则以其它文档的内容为准。

声明

我们已核对过本手册的内容与所描述的硬件和软件相符。由于差错难以完全避免，我们不能保证完全一致。我们会经常对手册中的数据进行检查，并在后续的版本中进行必要的更正。欢迎您提出宝贵意见。

西门子（中国）有限公司