

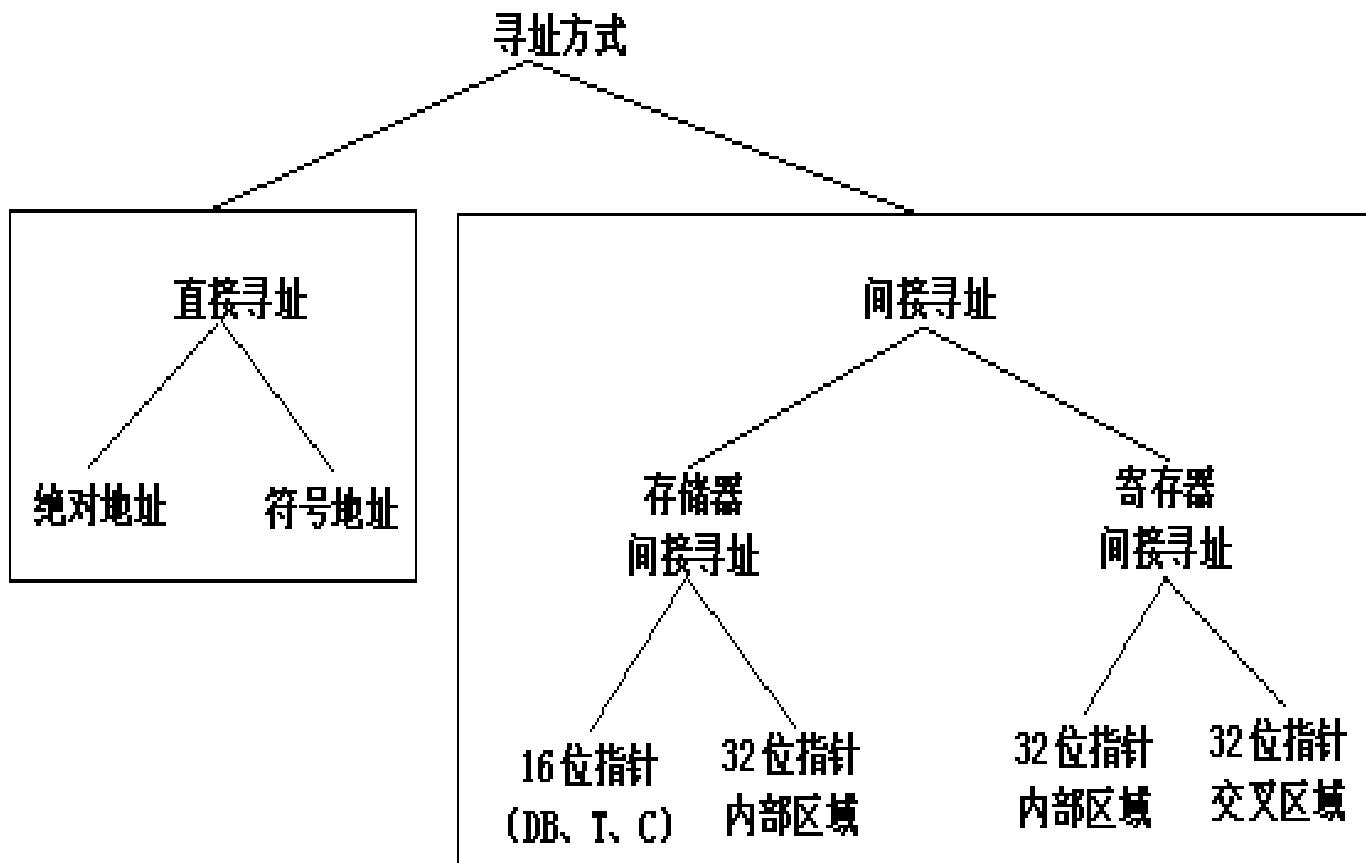


SIEMENS

STEP 7 指针编程

For internal use only

寻址方式
直接寻址
存储区地址指针
寄存器间接寻址
POINTER数据类型指针
ANY数据类型指针
FB块形参的编程



	地址区符号及访问长度	说明	举例
寻址方式	I/IB/IW/ID	过程映像区输入	I1.0、IB 2、IW4、ID12
直接寻址	Q/QB/QW/QD	过程映像区输出	Q 3.2、QB 12、QW20、QD40
存储区地址指针	PIB/PIW/PID	外设输入（或立即读）	PIB256、PIW300、PID400
寄存器间接寻址	PQB/PQW/PQD	外设输出（或立即写）	PQB256、PQW288、PQD300
POINTER数据类型指针	M/MB/MW/MD	标志位存储区	M 4.0、MB 3、MW12、MD42
ANY数据类型指针	L/LB/LW/LD	区域数据	L 2.2、LB 1、LW20、LW42
	T	定时器	T1
	C	计数器	C1
	FC/FB/SFC/SFB	程序块	FC1、SFC 67
	DBX/DBB/DBW/DBD	数据块（使用 OPN DB）*	DBX12.0、DBB20、DBW40、DBD100
	DIX/DIB/DIW/DID	数据块（使用 OPN DI）*	DIX12.0、DIB20、DIW40、DID100
	* DB 块的访问也可以直接带有 DB 号，例如 DB1.DBX20.0。		

FB块形参的编程

```

A    M1.1
AN   DB1.DBX12.0
=    Q1.2

```

寻址方式

直接寻址

存储区地址指针

寄存器间接寻址

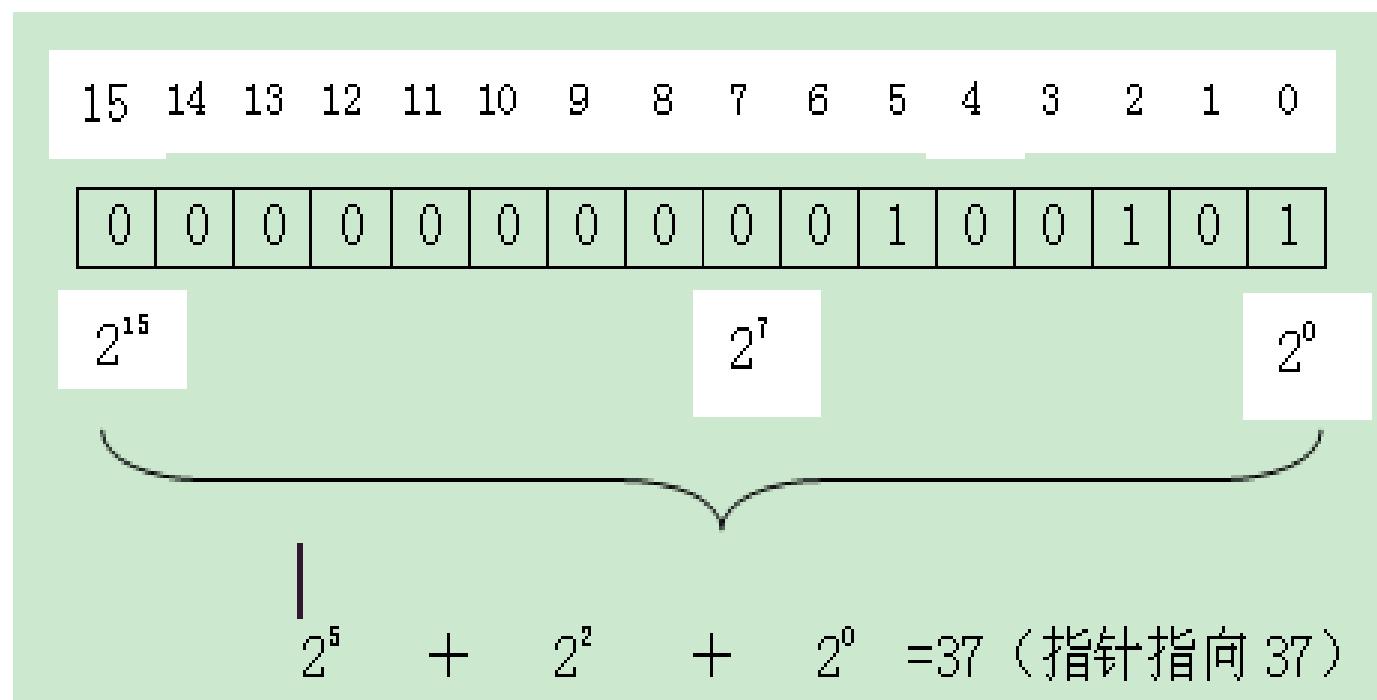
POINTER数据类型指针

ANY数据类型指针

FB块形参的编程

16位地址指针

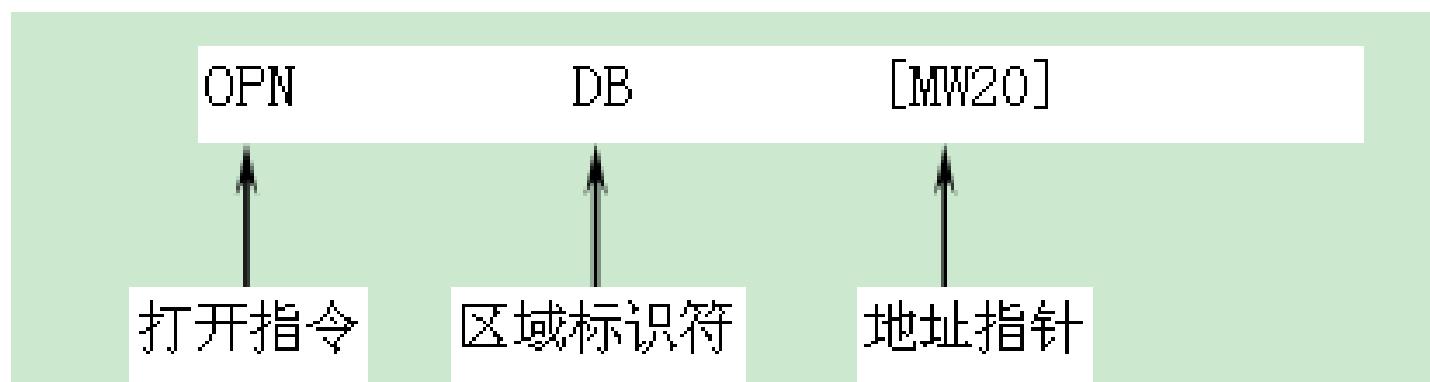
16位地址指针用于定时器、计数器、程序块（DB、FC、FB）的寻址，16位指针被看作一个无符号整数（0~65535），它表示定时器（T）、计数器（C）、数据块（DB、DI）或程序块（FB、FC）的号，16位指针的格式如下：



16位地址指针

- 寻址方式
- 直接寻址
- 存储区地址指针
- 寄存器间接寻址
- POINTER数据类型指针
- ANY数据类型指针
- FB块形参的编程

地址寻址表示格式为：区域标识符[16位地址指针]，例如打开一个DB块表示为：



16位地址指针使用示例

寻址方式

```
L    11          //将 11 传送到累加器 1 中。
T    MW    20      //将累加器 1 中的数值传送到 MW20 中。
A    I     2.1      //如果 I2.1 为 1，将预置值 10 秒装载到 T11 中。
```

直接寻址

```
L    S5T#10S
```

```
SE   T [MW 20]
```

存储区地址指针

```
L    MW    20
```

```
L    1
```

```
+I
```

```
T    MW    22      //MW20 再加 1。
```

```
A    I     2.2      //如果 I2.2 为 1，C12 向上计数一次。
```

POINTER数据类型指针

```
//////////////////////////////
```

```
L    12
```

```
T    LW    20
```

```
UC   FC [LW 20]    //无条件调用FC12
```

```
L    13
```

```
T    MW    20
```

```
A    I     2.3
```

```
CC   FB [MW 20]    //如果 I2.3 为 1，调用FB13。
```

FB块形参的编程

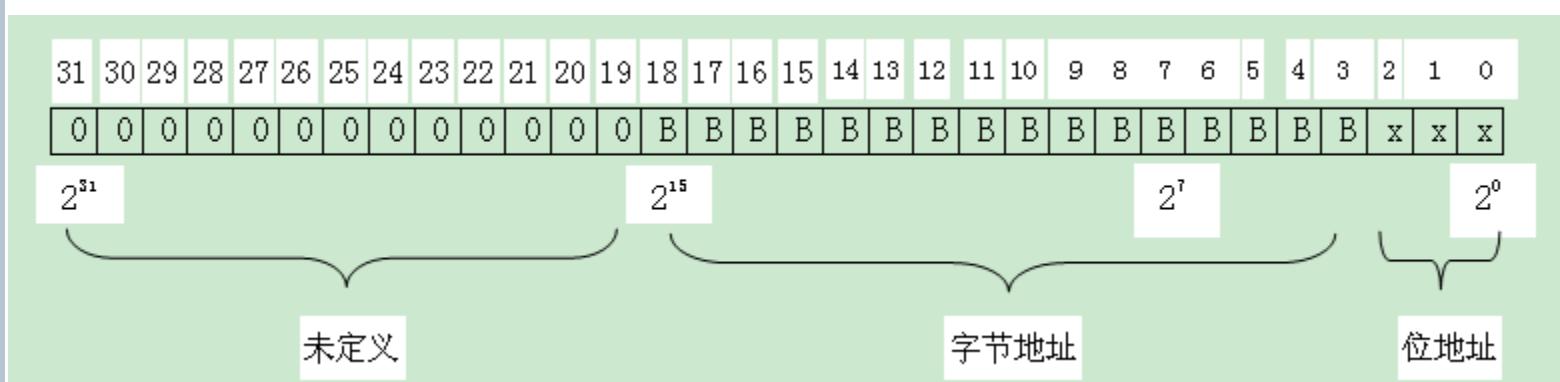
FC12和FB13不能带有形参，这是有CC和UC调用指令决定的。

32位地址指针

- 寻址方式
- 直接寻址
- 存储区地址指针
- 寄存器间接寻址
- POINTER数据类型指针
- ANY数据类型指针
- FB块形参的编程

32位地址指针

32位地址指针用于I、Q、M、L、数据块等存储器中位、字节、字及双字的寻址，32位的地址指针可以使用一个双字表示，第0位～第2位作为寻址操作的位地址，第3位～第18位作为寻址操作的字节地址，第19位～第31位没有定义，32位指针的格式如下：

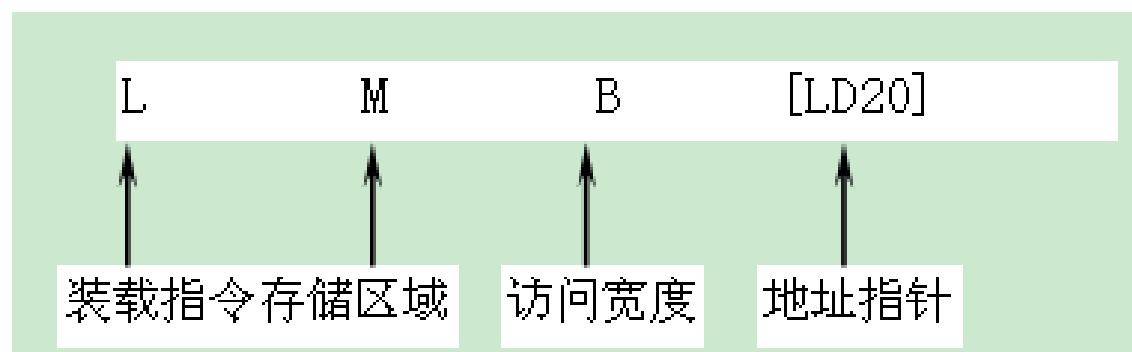


32位地址指针

- 寻址方式
- 直接寻址
- 存储区地址指针
- 寄存器间接寻址
- POINTER数据类型指针
- ANY数据类型指针
- FB块形参的编程

地址寻址表示格式为：

地址存储器标识符[32位地址指针]，例如指针存储于LD20中，装载M存储器一个字节表示



32位地址指针也可以使用常数表示，例如装载32位指针常数 **L P# 40.3** (**P=指针**，字节地址=40，位地址=3)。32位地址指针数据与双整数可以相互转换，由于指针指到一个位地址上，每一个位地址加1，相应转换的整数值加1的倍数，例如**P#0.0**转换双整数为**L#0**，**P#0.1**转换双整数为**L#1**，每一个字节地址加1，相应转换的整数值加8的倍数，例如**P#3.1**转换双整数为**L#25**.

32位地址指针使用示例1

寻址方式	OPN DB 1 //打开DB1。
直接寻址	OPN DI 3 //打开DB3，最多可以同时打开两个DB块。
存储区地址指针	L 4 //装载4到累加器1中。 SLD 3 //累加器1中数值左移3位。
寄存器间接寻址	T MD 20 //将逻辑操作结果传送到MD20中，MD20包含地址指针为P#4.0。
POINTER数据类型指针	L P#20.0 //将地址指针P#20.0装载到MD24中。 T MD 24 L 320 //320转换指针为P#40.0并装载到MD28中。 T MD 28 L DBW [MD 20] //装载DB1.DBW4。 L DBW [MD 24] //装载DB1.DBW20。
ANY数据类型指针	+I //相加 L DIW [MD 28] //装载DB3.DBW40。 -I //相减。
FB块形参的编程	T DIW 2 //将运算结果传送到DB3.DBW2中。 JC m1

32位地址指针使用示例2

- 寻址方式
- 直接寻址
- 存储区地址指针
- 寄存器间接寻址
- POINTER数据类型指针
- ANY数据类型指针
- FB块形参的编程

使用**LOOP** 指令与32位地址指针可以进行循环操作，假设一个编程应用：一个字变量(**MW2**)与一个数组（假设存储于**DB1**中，包含100个元素为字的数组）存储的值相比较，如果数值相同，指出第一个相同数值存储在**DB**块中的位置（数组中的位置）。

```
L 0          //初始化MW100和MD4。  
T MW 100  
T MD 4  
OPN DB 1    //打开DB1。  
L 100        //循环操作的次数，100次。  
next: T MW 100 //将循环100次装载到MW100中，  
                //固定格式。  
L MW 2      //进行比较的数值存储于MW2。  
L DBW [MD 4] //与DB块中存储的值进行比较,开  
                //始地址为DBW0。  
==|          //如果数值相等跳到m1。
```

32位地址指针使用示例2

寻址方式
直接寻址
存储区地址指针
寄存器间接寻址
POINTER数据类型指针
ANY数据类型指针
FB块形参的编程

```
JC m1
L MD 4          //将地址指针加2（每个相邻的字地址相差2）。
L P#2.0
+D
T MD 4
L MW 100        //次数减1，跳回next，如果MW100等于0，跳出循环操作LOOP指令，LOOP指令固定格式。
LOOP next
m1: FP M 10.0   //如果数值相当，记录MD4指针的数据，将转换为数组的位置((地址值/P#2.0)+1)值存储于MD8中。
JCN m2
L MD 4
L P#1.0
T MD 8
m2: NOP 0
```

32位地址指针注意事项

寻址方式
直接寻址
存储区地址指针
寄存器间接寻址
POINTER数据类型指针
ANY数据类型指针
FB块形参的编程

使用32位地址指针的注意事项：

```
OPN DB 1
L 20
T MD 20      //MD20装载的地址指针为P#2.4。
L 11
T DBB [MD 20] //指针指向P#2.4，相当于L
                  DBB2.4，CPU无法识别，将停机。
```

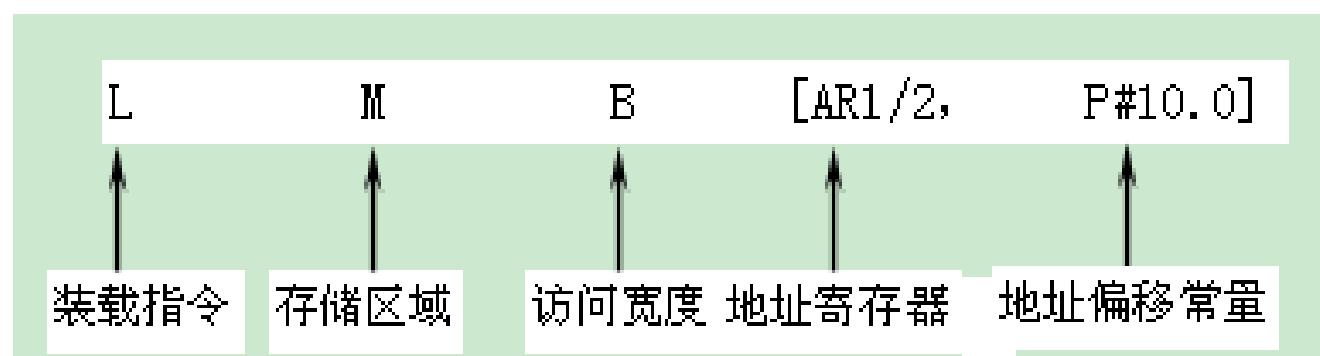
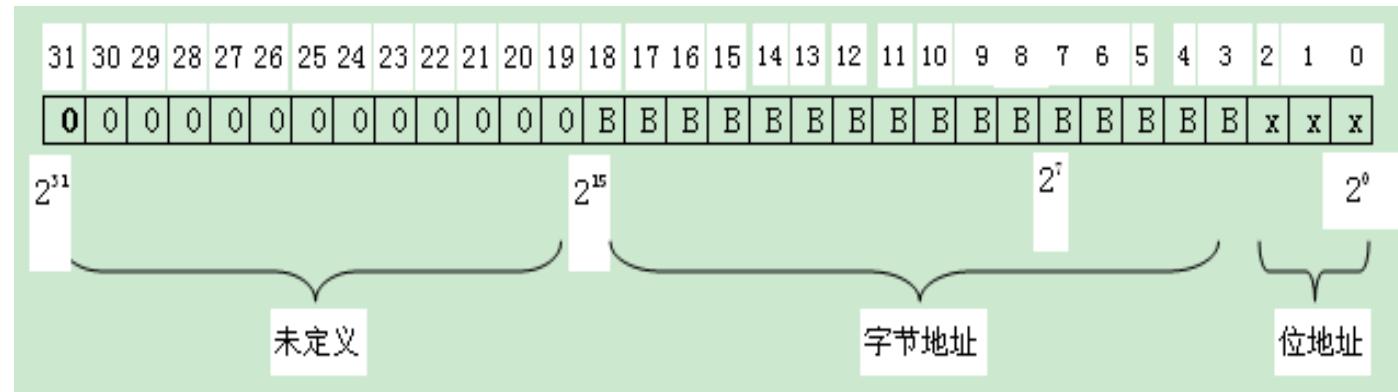
如果对相邻两个字节操作，指针转换为整数值最小必须为8(指针为P#1.0)的倍数，如果对相邻两个字操作，指针转换为整数值最小必须为16(指针为P#2.0)的倍数，，如果对相邻两个双字操作，指针转换为整数值最小必须为32(指针为P#4.0)的倍数，对字与双字指针的要求主要防治数据间的冲突，例如DBW[MD2]，MD2为16的倍数时，按照DBW2、DBW4、DBW6寻址，如果为8的倍数，按照DBW1、DBW2、DBW3寻址，地址间数据冲突。

寄存器间接寻址使用的指令

寻址方式	寄存器间接寻址使用CPU内部集成的两个32位寄存器AR1和AR2
直接寻址	
存储区地址指针	LAR1 : 将ACCU1存储的地址指针写入AR1。 LAR1 <D> : 将指明的地址指针写入AR1, 例如LAR1 P#20.0或 LAR1 MD20
寄存器间接寻址	LAR1 AR2 : 将AR2的内容写入AR1。 LAR2 : 将ACCU1存储的地址指针写入AR2。 LAR2 <D> : 将指明的地址指针写入AR2, 与LAR1 <D>方式相同。
POINTER数据类型指针	TAR1 : 将AR1存储的地址指针传输给ACCU1。 TAR1<D> : 将AR1存储的地址指针传输给指明的变量中。
ANY数据类型指针	TAR1 AR2 : 将AR1存储的地址指针传输给ACCU2。 TAR2 : 将AR2存储的地址指针传输给ACCU1。 TAR2 <D> : 将AR1存储的地址指针传输给指明的变量中。
FB块形参的编程	CAR : 交换AR1和AR2的内容。

32位内部区域指针

- 寻址方式
- 直接寻址
- 存储区地址指针
- 寄存器间接寻址
- POINTER数据类型指针
- ANY数据类型指针
- FB块形参的编程

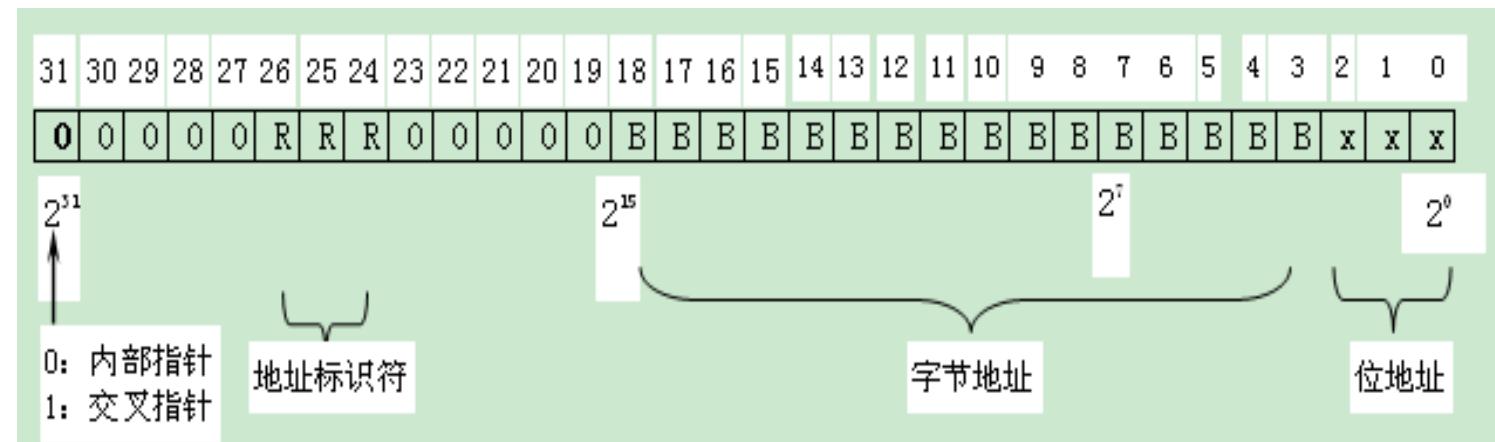


32位内部区域指针使用示例

寻址方式	OPN DB 1 //打开DB1。
直接寻址	LAR1 P#10.0 //将指针P#10.0 装载到地址寄存器1中。
存储区地址指针	L DBW [AR1,P#12.0] //将DBW22装载到累加器1中。
寄存器间接寻址	LAR1 MD 20 //将存储于MD20中的指针装载到地址寄存器1中。 L DBW [AR1,P#0.0] //将DBW装载到累加器1中,地址存储于MD20中。
POINTER数据类型指针	+ LAR2 P#40.0 //将指针P#40.0 装载到地址寄存器2中。 T DBW [AR2,P#0.0] //运算结果传送到DBW40中。
ANY数据类型指针	
FB块形参的编程	

32位交叉区域指针

- 寻址方式
- 直接寻址
- 存储区地址指针
- 寄存器间接寻址
- POINTER数据类型指针
- ANY数据类型指针
- FB块形参的编程

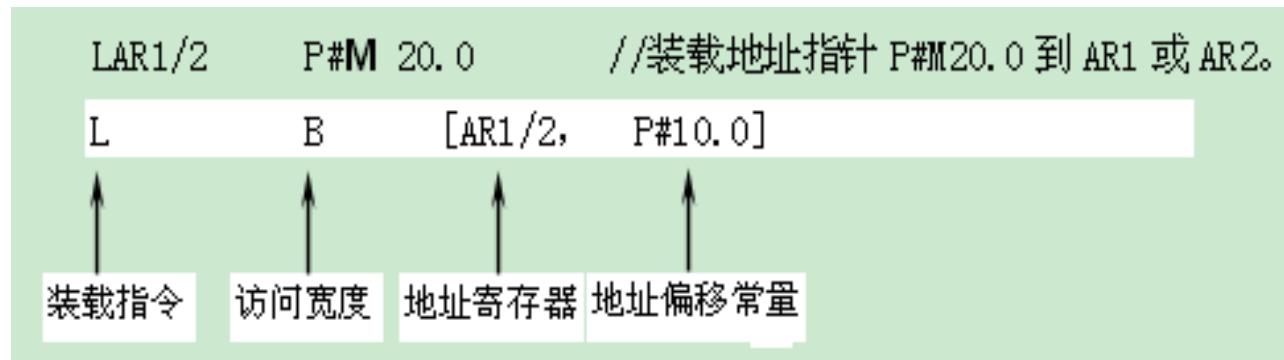


000表示没有地址区，例如P#12.0;
 001表示输入地址区I，例如P#I12.0;
 010表示输出地址区Q，例如P#Q12.0;
 011表示标志位地址区M，例如P#M12.0;
 100表示数据块(DB)中的数据，例如P#DB1.DBX12.0
 101表示数据块(DI)中的数据，例如P#DI1.DIX12.0
 110表示区域地址区L，例如P#L12.0;
 111表示调用程序块的区域地址区V，例如P#V12.0;

32位交叉区域指针使用示例

- 寻址方式
- 直接寻址
- 存储区地址指针
- 寄存器间接寻址
- POINTER数据类型指针
- ANY数据类型指针
- FB块形参的编程

使用交叉区域指针表示方法（例如装载M存储器一个字节）为：



```

LAR1 P#M 20.0      //将指针P#M20.0 装载到地址寄存器1中。
A   [AR1,P#1.1]      //M21.1“与”操作。
=   Q   1.2           //如果M21.1为1，输出1.2为1。
L   P#I 40.0          //将指针P#I40.0 装载到累加器1中。
LAR2                  //将累加器1中存储的地址指针装载到地址
                        寄存器2中。
L   W [AR2,P#0.0]     //装载IW40.0到累加器1中。
T   MW   60           //将累加器1中存储的数值传送到MW60中。

```

地址寄存器AR1、AR2的限制

寻址方式

直接寻址

存储区地址指针

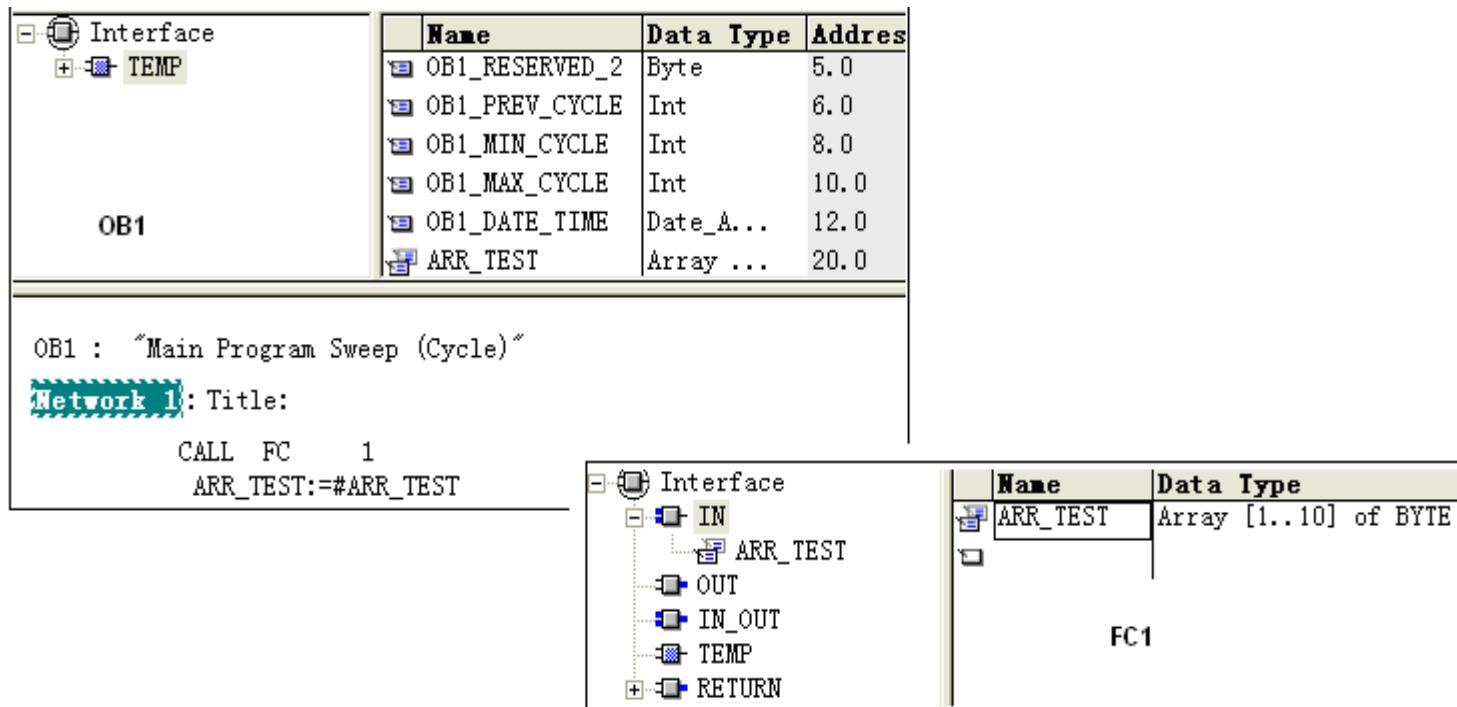
寄存器间接寻址

POINTER数据类型指针

ANY数据类型指针

FB块形参的编程

1:在形参的传递中，STEP7使用地址寄存器AR1访问函数FC接口及函数块FB“INOUT”接口中定义的复合类型参数，如ARRAY、STRUCT、DATE_AND_TIME等，AR1和DB块寄存器中的内容将被覆盖



地址寄存器AR1、AR2的限制

寻址方式

直接寻址

```
OPN DB 1
LAR1 P#20.0
L #ARR_TEST[1]
T DBW [AR1, P#0.0]
```

存储区地址指针

寄存器间接寻址

POINTER数据类型指针

ANY数据类型指针

FB块形参的编程

The diagram illustrates the memory map of a database (DB1) with two accumulators (AR1, AR2). The database structure is as follows:

	AR 1	AR 2	DB1
	0.0	0.0	1
	20.0	0.0	1
V	20.0	0.0	0
V	20.0	0.0	0

避免方法:

```
L #ARR_TEST[1]          //装载形参变量ARR_TEST[1]到累加器1中。
OPN DB 1                //打开OB1
LAR1 P#20.0              //将P#20.0装载到地址寄存器AR1中。
T DBW [AR1,P#0.0]        //将累加器1中的值传送到DB1.DBW20中。
```

地址寄存器AR1、AR2的限制

寻址方式

直接寻址

存储区地址指针

寄存器间接寻址

POINTER数据类
型指针

ANY数据类型指
针

FB块形参的编程

2: AR2和DI寄存器分别包含FB背景数据块的块号及在背景数据块中偏移地址（多重背景数据块），在FB中使用AR2和DI寄存器将会覆盖系统存储的内容。

```
TAR2 MD 100      //将AR2的数据存储于MD100中。  
L DINO          //将背景DB块块号存储于MW104中。  
T MW 104
```

/////////用户程序//////////

```
LAR2 MD 100      //将MD100中存储的地址指针装载到AR2  
                  中。  
OPN DI [MW 104]  //打开DI数据块。
```

3: LAR1 P##PARA(参数)。非法指令

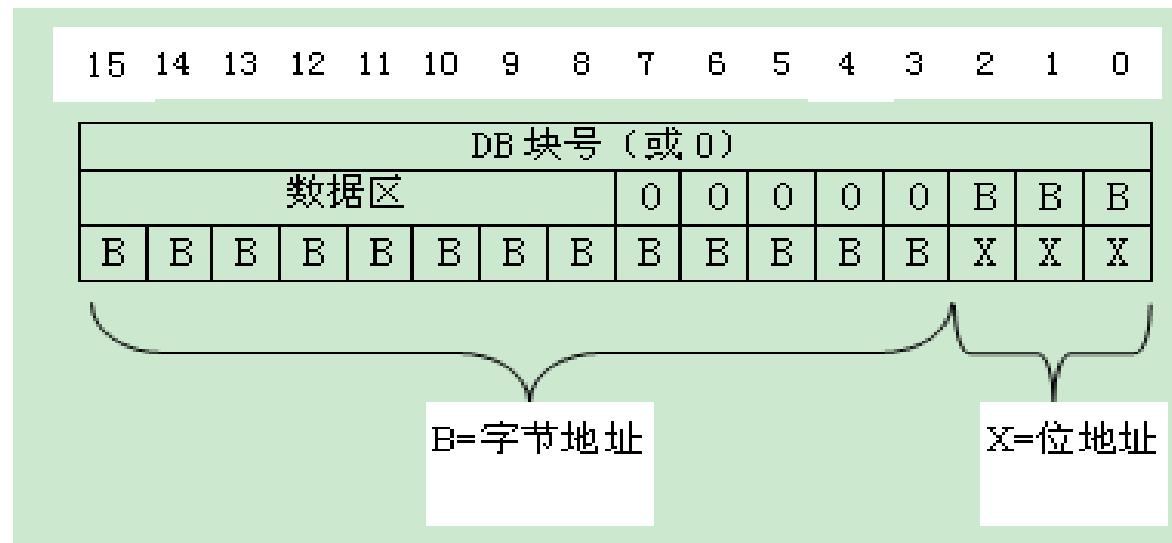
```
L P##PARA(参数)  //将地址指针装载到累加器1。  
LAR1/2
```

POINTER数据类型指针

- 寻址方式
- 直接寻址
- 存储区地址指针
- 寄存器间接寻址
- POINTER数据类型指针
- ANY数据类型指针
- FB块形参的编程

POINTER数据类型指针用于向被调用的函数FC及函数块FB传递复杂数据类型（如ARRAY、STRUCT及DT等）的实参。在被调用的函数FC及函数块FB内部可以间接访问实参的存储器。

POINTER指针占用48位地址空间，数据格式如下：



POINTER数据类型指针

寻址方式	
直接寻址	
存储区地址指针	
寄存器间接寻址	
POINTER数据类型指针	
ANY数据类型指针	
FB块形参的编程	

POINTER指针数据区的表示：

16进制代码	数据区	简单描述
B#16#81	I	输入区
B#16#82	Q	输出区
B#16#83	M	标志位
B#16#84	DB	数据块
B#16#85	DI	背景数据块
B#16#86	L	区域数据区
B#16#87	V	上一级赋值的区域数据

POINTER数据类型指针表示方法,例如:

P# DB2.DBX12.0 //指向DB2.DBX12.0。

P#M12.1 //指向M12.1。

也可以选择使用地址声明或符号名（不使用符号P#）的方式进行赋值，例如：

DB2.DBX12.0 //指向DB2.DBX12.0。

M12.1 //指向M12.1。

POINTER数据类型指针使用示例

- 寻址方式
- 直接寻址
- 存储区地址指针
- 寄存器间接寻址
- POINTER数据类型指针
- ANY数据类型指针
- FB块形参的编程

编写一个计算功能的函数FC3，输入首地址“In_Data”及连续浮点格式变量的个数“NO”后，输出几个变量的平均值“OUT_VAL”。
OB1中调用函数FC3的程序如下：

```
CALL FC 3          // 调用函数3。
In_Data:=P#M 100.0 //输入的首地址。
NO      :=4         //变量的个数。
OUT_VAL:=MD20      //计算结果。
```

完成的计算功能相当于 $MD20 := (MD100+MD104+MD108+MD112) / 4$

FC3接口参数

数据接口	名称	数据类型	地址
IN	In_Data	Pointer	
IN	NO	INT	
OUT	OUT_VAL	REAL	
TEMP	BLOCK_NO	INT	0.0
TEMP	NO_TEMP	INT	2.0
TEMP	ADD_TEMP	REAL	4.0

POINTER数据类型指针使用示例

寻址方式

```
L    0           //初始化临时变量#ADD_TEMP。
T    #ADD_TEMP
L    P##In_Data //指向存储地址指针P#M100.0的首地址，并装载到地址寄存器AR1中。
```

直接寻址

存储区地址指针

```
LAR1
L    0           //判断OB1中赋值的地址指针是否为数据块（参考POINTER的数据格式）。
```

寄存器间接寻址

```
L    W [AR1, P#0.0]
==I
JC   M1
T    #BLOCK_NO
```

POINTER数据类型指针

```
M1: OPN  DB [#BLOCK_NO] //如果是DB块，打开指定的DB块。
      L    D [AR1, P#2.0] //找出需要计算数据区的开始地址，  
POINTER数据中，后4个字节包含内部  
交叉指针，将  
P#M100.0装载到AR1中。
```

ANY数据类型指针

```
LAR1
L    0           //如果输入变量个数为0，结束FC3的执行。如果不等于0作为循环执行的次数（NO_TEMP）。
```

FB块形参的编程

```
L    #NO
==I
JC   END
```

POINTER数据类型指针使用示例

寻址方式

直接寻址

存储区地址指针

寄存器间接寻址

POINTER数据类
型指针

ANY数据类型指
针

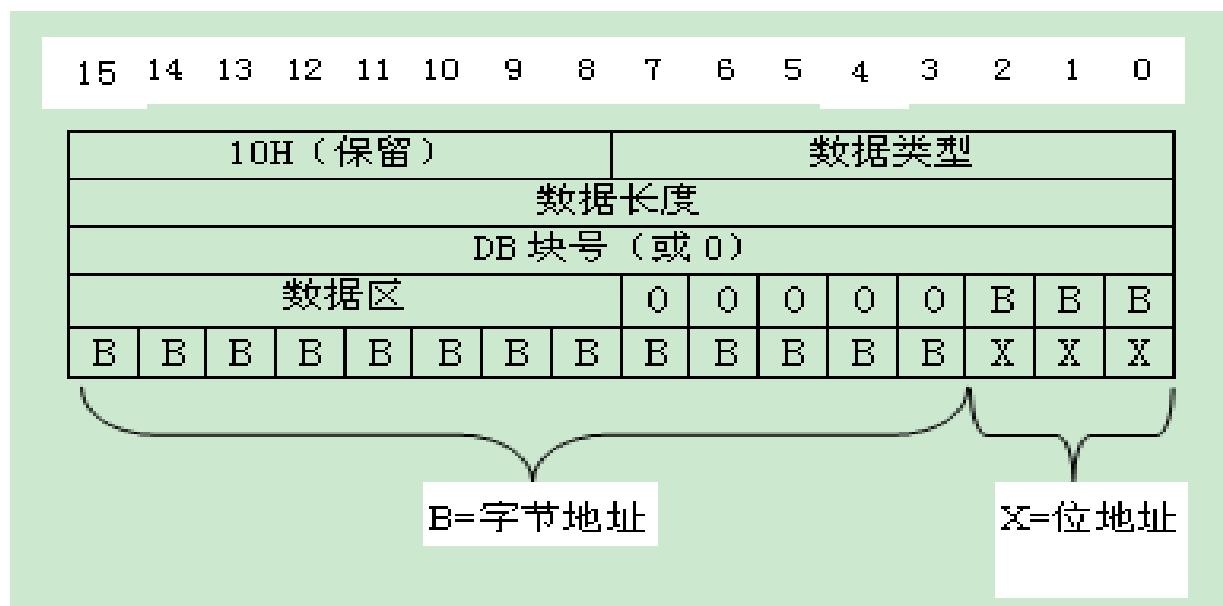
FB块形参的编程

```
NO:    T      #NO_TEMP           //循环执行加运算，本例中循环执行的次  
                  //数为4。  
          L      D [AR1, P#0. 0]     //装载MD100到累加器1中。  
          L      #ADD_TEMP         //与临时变量#ADD_TEMP相加后将计算结  
          +R                果再存储于#ADD_TEMP中。  
          T      #ADD_TEMP         //地址寄存器加4, 下一次于MD104相加。  
          +AR1   P#4. 0            //LOOP 指令固定格式。  
          L      #NO_TEMP          //跳回“NO”循环执行，执行完定义在变  
          LOOP   NO               量#NO_TEMP的次数后自动跳出循环程序。  
          L      #ADD_TEMP         //求平均值，装载运算结果到累加器1中。  
          L      #NO               //将变量个数转变为浮点值便于运算。  
          DTR              /R  
          T      #OUT_VAL          //输出运算结果。  
END:   NOP   0
```

ANY数据类型指针

- 寻址方式
- 直接寻址
- 存储区地址指针
- 寄存器间接寻址
- POINTER数据类型指针
- ANY数据类型指针
- FB块形参的编程

ANY数据类型指针中包括数据类型、重复系数、DB块号、存储器机数据开始地址，占用80位地址空间，数据格式如下：



ANY指针数据区的表示

数据类型代码			
寻址方式	十六进制代码	数据类型	简单描述
直接寻址	B#16#00	NIL	空
	B#16#01	BOOL	位
	B#16#02	BYTE	8位字节
	B#16#03	CHAR	8位字符
存储区地址指针	B#16#04	WORD	16位字
	B#16#05	INT	16位整形
	B#16#06	DWORD	32位双字
	B#16#07	DINT	32位双整形
	B#16#08	REAL	32位浮点
寄存器间接寻址	B#16#09	DATE	IEC日期
	B#16#0A	TIME_OF_DAY (TOD)	24小时时间
POINTER数据类型指针	B#16#0B	TIME	IEC时间
	B#16#0C	S5TIME	SIMATIC时间
	B#16#0E	DATE_AND_TIME (DT)	时钟
ANY数据类型指针	B#16#13	STRING	字符串
	B#16#17	BLOCK_FB	FB号
	B#16#18	BLOCK_FC	FC号
FB块形参的编程	B#16#19	BLOCK_DB	DB号
	B#16#1A	BLOCK_SDB	SDB号
	B#16#1C	COUNTER	计数器
	B#16#1D	TIMER	定时器

ANY数据类型指针

- 寻址方式
- 直接寻址
- 存储区地址指针
- 寄存器间接寻址
- POINTER数据类型指针
- ANY数据类型指针
- FB块形参的编程

与POINTER指针相比，ANY类型指针可以表示一段长度的数据，例如 P#DB1. DBX0. 0 BYTE 10，表示指向DB1. DBB0～DB1. DBB9。调用FB、FC时，对POINTER数据类型参数进行赋值时可以选择指针显示方式直接赋值，例如：

P# DB2. DBX12. 0 WORD 22 //指向从DB2. DBW12开始22个字。
P#M12. 1 BOOL 10 //指向从M12. 1开始10个位信号。

也可以选择使用地址声明或符号名（不使用符号P#）的方式进行赋值，例如：

DB2. DBW12 //指向DB2. DBW12一个字，数据长度为1。
M12. 1 //指向M12. 1一个位信号，数据长度为1。

ANY数据类型指针使用示例

- 寻址方式
- 直接寻址
- 存储区地址指针
- 寄存器间接寻址
- POINTER数据类型指针
- ANY数据类型指针
- FB块形参的编程

编写一个计算功能的函数FC13，输入参数“`In_Data`”为一个数组变量，如果数组元素为浮点数，输出所有元素的平均值“`OUT_VAL`”，如果数组元素为其它数据类型，不执行计算功能。OB1中调用函数FC13的程序如下：

```
CALL FC    13          // 调用函数13。
In_Data:=P#DB1. DBX0. 0 REAL 8  //输入数据区从DB1. DBD0开始8个浮点
                                 值。
OUT_VAL:=MD20           //计算结果。
完成的计算功能相当于MD20:=(DB1. DBD0+..+..+DB1. DBD28)/8。
```

FC13接口参数

数据接口	名称	数据类型	地址
IN	In_Data	ANY	
OUT	OUT_VAL	REAL	
TEMP	DATA_LEN	INT	0. 0
TEMP	BLOCK_NO	INT	2. 0
TEMP	ADD_TEMP	REAL	4. 0
TEMP	DATA_NO	INT	8. 0

ANY数据类型指针使用示例

寻址方式
直接寻址
存储区地址指针
寄存器间接寻址
POINTER数据类型指针
ANY数据类型指针
FB块形参的编程

```

L   0           //初始化临时变量#ADD_TEMP。
T   #ADD_TEMP
L   P##In_Date //指向存储地址指针In_Date首地址，并
                装载到地址寄存器AR1中。
LAR1
L   B [AR1, P#1. 0] //如果数据类型不是REAL，跳转到END。
L   B#16#8
<>R
JC  END
L   0
L   W [AR1, P#4. 0] //判断OB1中赋值的地址指针是否为数据
                    块（参考 ANY的数据格式）。
==I
JC  M1
T   #BLOCK_NO
OPN DB [#BLOCK_NO] //如果是DB块，打开指定的DB块。
M1: L   W [AR1, P#2. 0] //判断ANY指针中数据长度，本例中为
                    REAL变量的个数。
T   #DATA_LEN
L   D [AR1, P#6. 0] //找出需要计算数据区的开始地址，本例
                    中为DB1. DBX0. 0。
LAR1

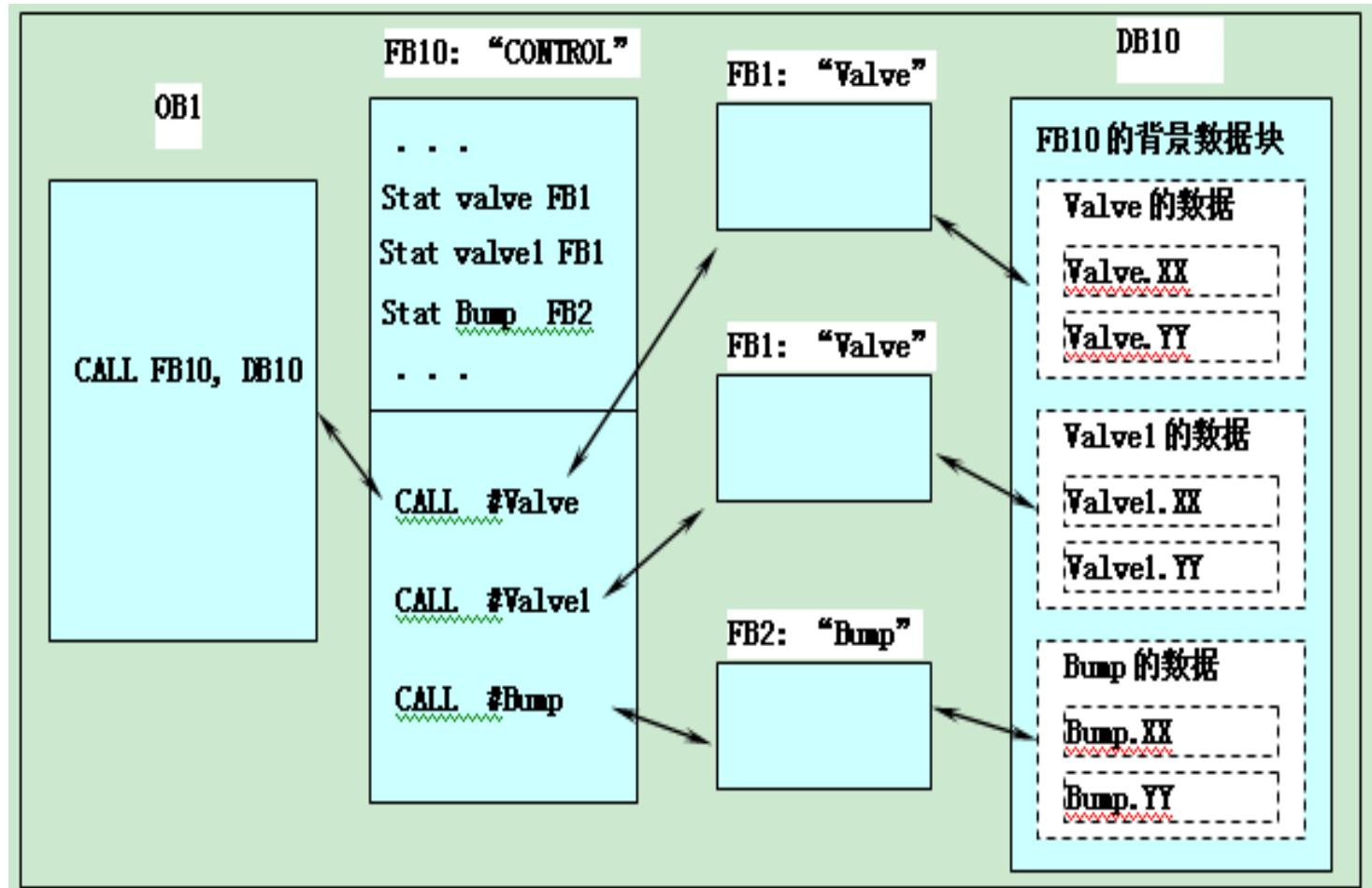
```

ANY数据类型指针使用示例

寻址方式	L #DATA_LEN	
直接寻址	NO: T #DATA_NO	//循环执行加运算，本例中循环执行的次数为8。
存储区地址指针	L D [AR1, P#0.0]	//装载DB1. DBD0到累加器1中。
	L #ADD_TEMP	//与临时变量#ADD_TEMP相加后将计算结果再存储 #ADD_TEMP中。
寄存器间接寻址	+R	
	T #ADD_TEMP	
	+AR1 P#4.0	//地址寄存器加4，地址偏移量。
POINTER数据类型指针	L #DATA_NO	//LOOP 指令固定格式。
	LOOP NO	//跳回“NO”循环执行，执行完定义在变量#NO_TEMP的次数后自动跳出循环程序。
	L #ADD_TEMP	//求平均值，装载运算结果到累加器1中。
ANY数据类型指针	L #DATA_LEN	
	DTR	//将变量个数转变为浮点值便于运算。
	/R	
	T #OUT_VAL	//输出运算结果。
FB块形参的编程	END: NOP 0	

FB块在多重数据块中的寻址

- 寻址方式
- 直接寻址
- 存储区地址指针
- 寄存器间接寻址
- POINTER数据类型指针
- ANY数据类型指针
- FB块形参的编程



FB块在多重数据块中的寻址

寻址方式
直接寻址
存储区地址指针
寄存器间接寻址
POINTER数据类型指针
ANY数据类型指针
FB块形参的编程

如果在FB1、FB2中使用POINTER或ANY数据类型指针进行拆分时，不考虑在多重背景DB块中的位置，将会造成错误，例如在FB1中定义输入接参数FB1_POS，数据类型为POINTER，在FB1中的程序如下：

```
L P##FB1_POS          //指向存储地址指针FB1_POS首地址。  
LAR1                  //存储于地址寄存器1中。  
L D [AR1, P#2.0]      //装载实参赋值的地址指针，并传送到  
                      MD20中。
```

T MD 20

同样在FB2中定义输入接参数FB2_POS，数据类型为POINTER，在FB2中的程序如下：

```
L P##FB2_POS          //指向存储地址指针FB2_POS首地址。  
LAR1                  //存储于地址寄存器1中。  
L D [AR1, P#2.0]      //装载实参赋值的地址指针，并传送到  
                      MD24中。
```

T MD 24

FB块在多重数据块中的寻址

- 寻址方式
- 直接寻址
- 存储区地址指针
- 寄存器间接寻址
- POINTER数据类型指针
- ANY数据类型指针
- FB块形参的编程

在FB10中，将FB1、FB2作为静态变量使用，FB10的接口参数为

数据接口	名称	数据类型	地址
STAT	FB1_POS	FB1	0.0
STAT	FB2_POS	FB2	6.0

FB10的程序如下：

```
CALL #FB1_POS          //调用FB1，赋值地址指针P#M100.0。  
FB1_POS:=P#M 100.0
```

```
CALL #FB2_POS          //调用FB2，赋值地址指针P#M120.0。  
FB2_POS:=P#M 120.0
```

在OB1中调用FB10，并生成DB10，程序如下：

```
CALL FB 10 , DB10 //调用FB10，生成DB10。
```

FB块在多重数据块中的寻址

- 寻址方式
- 直接寻址
- 存储区地址指针
- 寄存器间接寻址
- POINTER数据类型指针
- ANY数据类型指针
- FB块形参的编程

解决办法：

```
T AR2          //将偏移地址传送到累加器1中。  
L   DW#16#FFFFFF //过滤地址区，如将P#M20.0变为P#20.0。  
AD  
L   P##FB1_POS //将偏移地址与FB1_POS首地址相加并装载到  
                  AR1中。  
+D  
LAR1          //得到FB1在多重背景DB块中的首地址。  
L   D [AR1, P#2.0] //将P#M100.0装载到MD20中。  
T   MD    20
```



SIEMENS

Thank you

Name: SLC I IA&DT CS

Department: 西门子（中国）有限公司工业业务领域工业自动化与驱动技术集团技术支持与服务部

Phone: 4008104288

For internal use only