1 西门子自动化与驱动产品的在线技术支持

首先，建议您访问西门子 A&D 的产品与技术支持网站，

http://support.automation.siemens.com/CN/llisapi.dll?func=cslib.csinfo2&aktprim=99&lang=zh 主页如下所示：

您还可以访问西门子（中国）自动化与驱动集团的主页 www.ad.siemens.com.cn ，如下图所示。该网站同样可以提供大量的产品和系统的信息。

2　　如何获得西门子自动化与驱动产品的资料

首先，建议您通过 Siemens A&D 的网站搜索并下载。

您还可以致电 010-64721888 转 3785 /3726 索取资料。

另外，还有大量的手册可以通过分销商订购，和其它产品的订货方式一样。

3　　需设备选型及订货

如需设备选型及订货，请联系西门子自动化与驱动的销售工程师或当地西门子分销商。分销商联系方式可致电 400-810-4288 听到提示音后按 3 获得。

4　　西门子技术支持热线

如有无法自行解决的技术问题，请拨打西门子技术支持热线 400-810-4288 登记，等待西门子技术支持工程师回复。我们会在 2 小时内予以响应。

请注意在登记问题时尽量准确地描述所使用产品的类型，以便尽快得到负责该产品的工程师的帮助。

技术支持传真：010-64719991。

技术支持邮箱：4008104288.cn@siemens.com。

5 西门子自动化产品的其它网站

www.s7-200.com 提供 S7-200 PLC 相关知识及软件下载。

https://mall.automation.siemens.com/CN/guest/查找西门子自动化与驱动的所有产品订货号、图片、及技术参数。

6 S7-300/S7-400 PLC 相关手册

- <S7-300 CPU 31xC 和 CPU 31x，技术数据>，请点击 http://www.ad.siemens.com.cn/download/Info/00001/1022.pdf 下载
- "S7-300 Programmable Controller, Hardware and Installation" 手册，请点击 http://www4.ad.siemens.de/WW/view/en/15390415 下载
- < S7-300 模块数据> 手册，请点击 http://support.automation.siemens.com/CN/llisapi.dll/csfetch/8859629/s7300_baugruppendaten_zh-CHS.pdf?func=cslib.csFetch&nodeid=22273000&forcedownload=true 下载
- “S7-300 Instruction List CPU 31xC, CPU 31x, IM 151-7 CPU, BM 147-1 CPU, BM 147-2 CPU”，请点击 http://support.automation.siemens.com/CN/llisapi.dll/csfetch/13206730/s7300_operation_list_chs.pdf?func=cslib.csFetch&nodeid=22722994&forcedownload=true 下载
- “Automation System S7-400 CPU Specifications” 手册，请点击 http://support.automation.siemens.com/CN/llisapi.dll/csfetch/23904550/CPU-Data_zh-CHS.pdf?func=cslib.csFetch&nodeid=24782352&forcedownload=true 或 http://www.ad.siemens.com.cn/download/Info/00001/N0279.pdf 下载
- “Automation System S7-400 Hardware and Installation” 手册，请点击 http://www.ad.siemens.com.cn/download/Info/00001/1011.pdf 或 http://support.automation.siemens.com/WW/llisapi.dll/csfetch/1117849/424ish_e.pdf?func=cslib.csFetch&nodeid=1139865&forcedownload=true (英文)下载
- “Automation System S7-400 Module Specifications”，请点击 http://support.automation.siemens.com/CN/llisapi.dll/csfetch/1117740/425rfh_ch.pdf?func=cslib.csFetch&nodeid=22589080&forcedownload=true 下载
- “S7-400 Instruction List CPU 412, 414, 416, 417” 手册，请点击 http://support.automation.siemens.com/CN/llisapi.dll/csfetch/23904435/OP-Liste_chs.pdf?func=cslib.csFetch&nodeid=24479435&forcedownload=true 下载

7 缩写词含义

IEC：International Electrotechnical Commission，国际电工技术委员会

FAQ：Frequently Asked Questions，常见问题解答

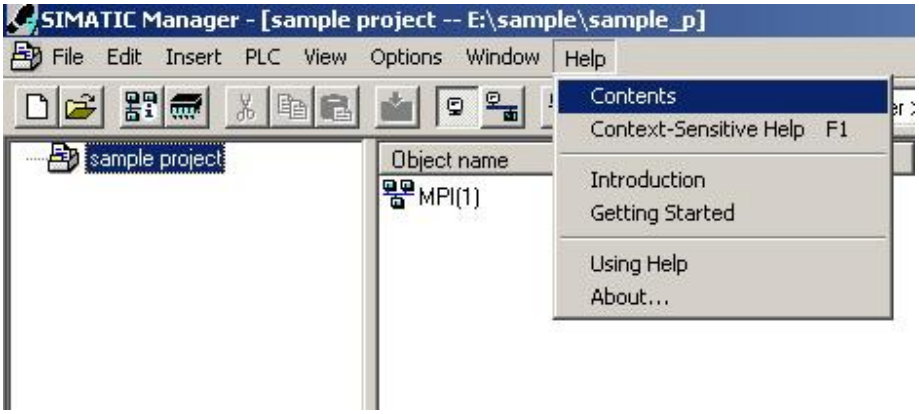
AVC：SIMATIC Card / Automation Value Card

8 如何使用 STEP 7 软件的在线帮助

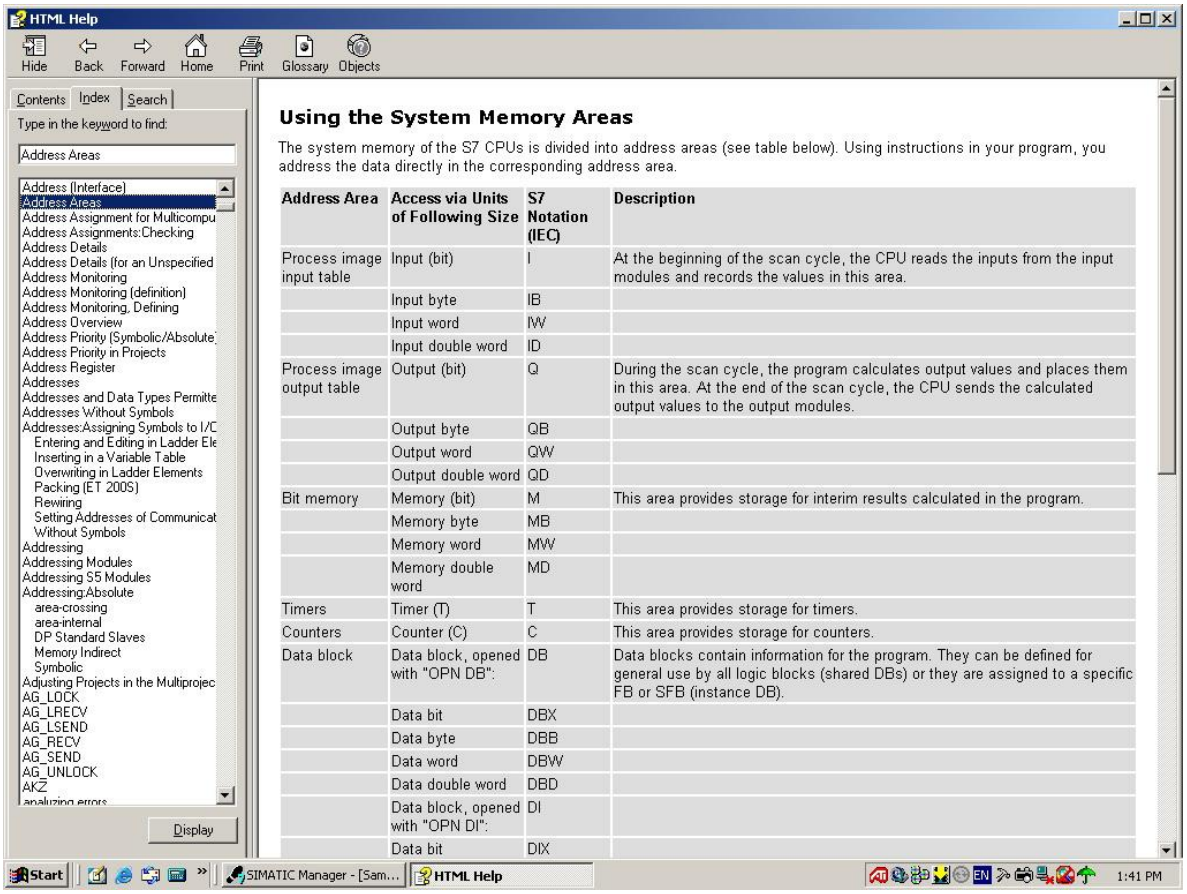
8.1 查找某个关键字或功能

8.1.1 在线帮助

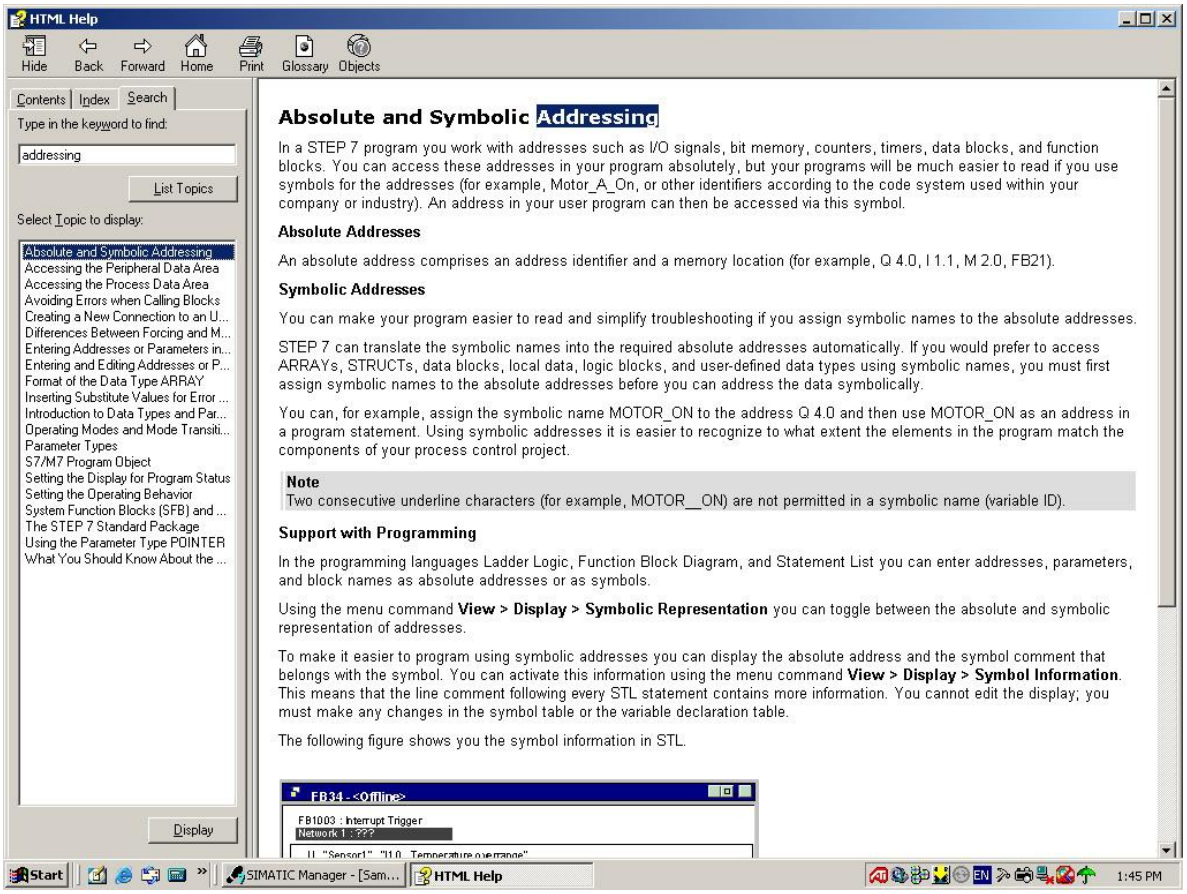
在 STEP 7 的主界面 SIMATIC Manager 中，点击下拉菜单 Help 下的 Contents，打开 STEP 7 的在线帮助



8.1.2 利用 Index 进行关键字的查找

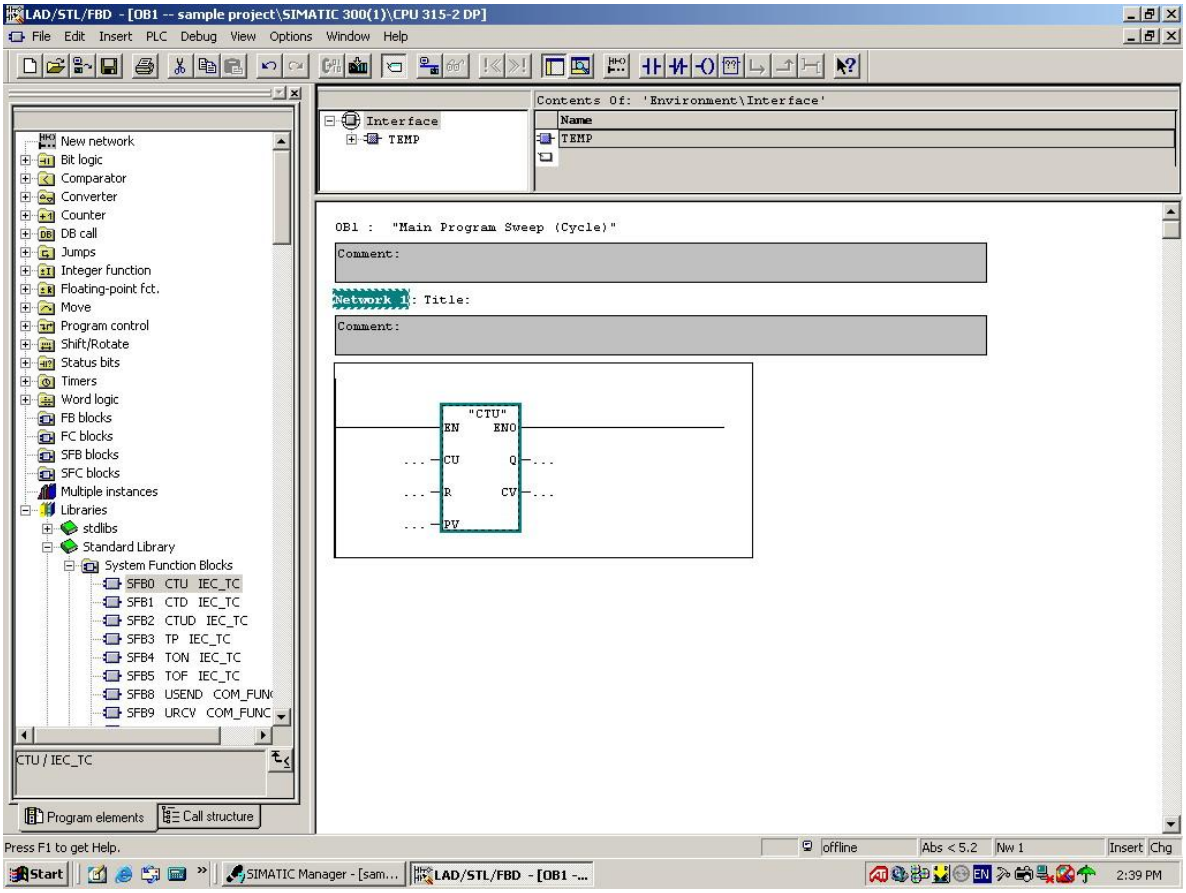


8.1.3　　或者，利用 Search 进行相关搜索

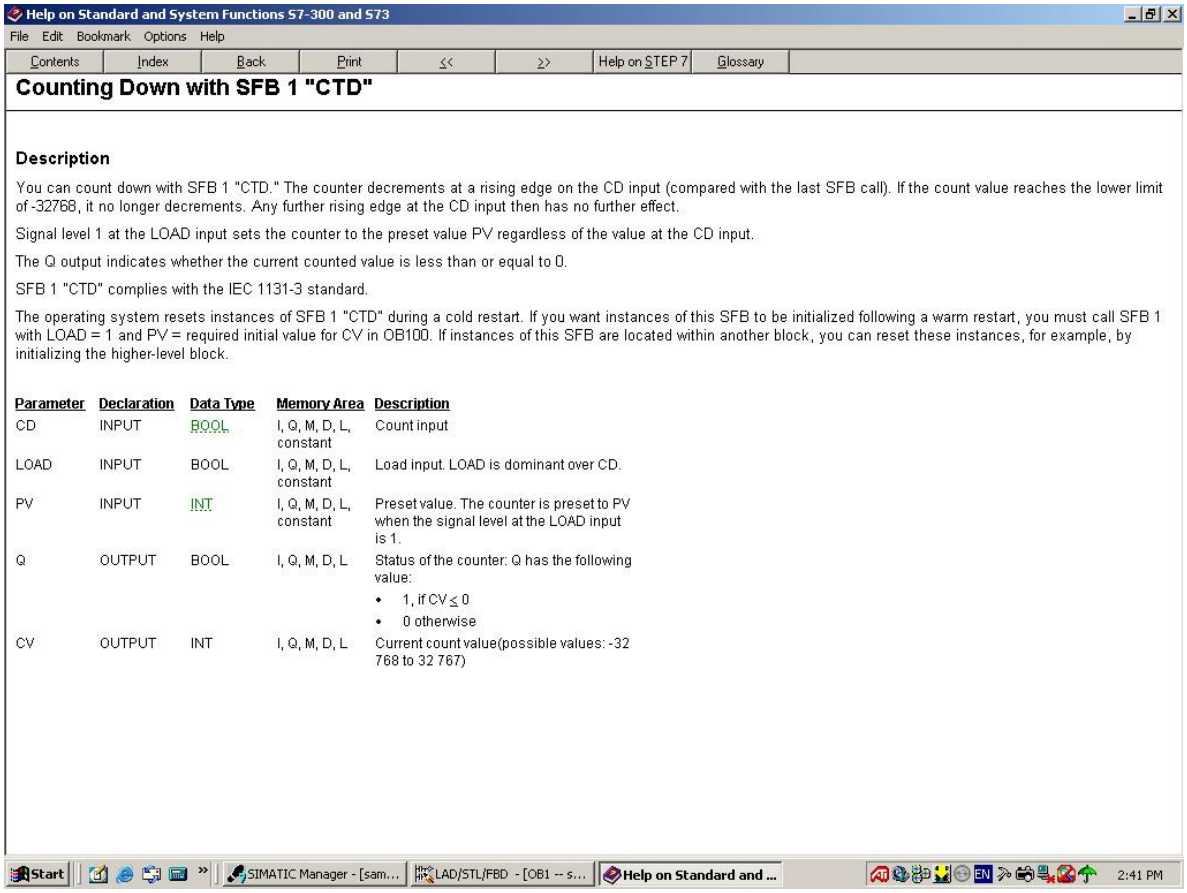


8.2　　想了解某个 FB/FC 的功能及管脚的定义

将您想要调用的 FB/FC 调入到一个 Network 中，选中该 FB/FC（用鼠标点击该 FB/FC，外框变为绿色），如下图所示：



按您计算机键盘上的 F1 功能键，就会跳出关于该 FB/FC 的功能及管脚定义的描述。如下图所示：



您可以在该帮助信息中了解到该 SFB/SFC/FB/FC 的功能、参数的描述及所要求的数据类型、可能的错误信息等，有些还有例子程序。

9 S7-300/400 系统存储区域

S7 CPU 的系统存储区域分为下表中列出的地址区域。在程序中可以根据相应的地址直接读取数据。

地址区域	可以访问的地址单位	S7 符号 (IEC)	描述
过程映像输入表	输入（位）	I	循环扫描周期开始时，CPU 从输入模板读输入值并记录到该区域
	输入（字节）	IB	
	输入（字）	IW	
	输入（双字）	ID	
过程映像输出表	输出（位）	Q	在循环扫描周期中，程序计算输出值并记录到该区域。循环扫描周期结束时，CPU 将计算结果写入相应的输出模板
	输出（字节）	QB	
	输出（字）	QW	
	输出（双字）	QD	
位存储器	存储器（位）	M	该区域用于存储程序的中间计算结果
	存储器（字节）	MB	

	存储器（字）	MW	
	存储器（双字）	MD	
定时器	定时器 (T)	T	该区域提供定时器的存储
计数器	计数器 (C)	C	该区域提供计数器的存储
数据块	数据块，用"OPN DB"打开	DB	数据块中包含了程序的信息。可以定义为所有逻辑块共享 (shared DBs) 或指定给一个特定的 FB 或 SFB 做背景数据块(instance DB)。
	数据位	DBX	
	数据字节	DBB	
	数据字	DBW	
	数据双字	DBD	
	数据块，用"OPN DI"打开	DI	
	数据位	DIX	
	数据字节	DIB	
	数据字	DIW	
	数据双字	DID	
局部数据	局部数据位	L	该区域包含块执行时该块的临时数据。L 堆栈还提供用于传递块参数及记录梯形逻辑网络中间结果的存储器
	局部数据字节	LB	
	局部数据字	LW	
	局部数据双字	LD	
外设地址 (I/O)	外设输入字节	PIB	主站及分布式从站（DP）外设输入输出区域允许直接存取
输入			
	外设输入字	PIW	
	外设输入双字	PID	
外设地址 (I/O)	外设输出字节	PQB	
输出			
	外设输出字	PQW	
	外设输出双字	PQD	

10 S7-300/400 PLC 寻址方式

10.1 直接寻址

在 STEP7 程序中你可以使用输入输出信号(I/O)，位存储区(M)，计数器，计时器，数据块(DB)以及功能块(FB)等地址。你可以直接访问这些绝对地址，但是如果给绝对地址以符号(助记符)程序将更易读懂(例如 Motor_A_On, 或者根据你的公司或者工程中的代码使用别的标识符),而一个你的用户程序中的地址也就可以用一个符号来访问。

10.1.1 绝对地址：

绝对地址由一个地址标识符和存储器位置组成。

例如 I 0.0，Q 1.7，PIW 256，PQW 512，MD 20，T 15，C 16，DB1.DBB 10，L1 0.0 等

10.1.2 符号寻址：

如果给绝对地址分配符号可使程序易读而简化故障查找。

STEP7 能自动翻译符号名为要求的绝对地址。如果你准备用符号名访问 数组，结构，数据块，局部变量，逻辑块及用户自定义数据类型，那么你必须在此之前先分配符号名给绝对地址。

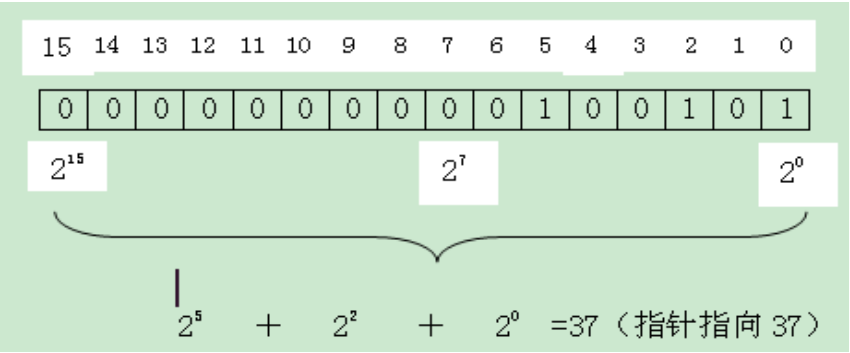
例如， 你可以分配符号名'Motor_On'给地址 Q 4.0，然后在程序语句中使用符号名'Motor_On'作为地址。

10.2 间接寻址

间接寻址分为存储器间接寻址和寄存器间接寻址，间接寻址的指针分为 16 位指针和 32 位指针，而 32 位指针又分为内部区域寻址与交叉区域寻址。

10.2.1 存储器间接寻址

*16 位指针：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：



寻址格式表示为： 区域标识符 [16 位地址指针]

例如使一个计数器向上计数表示为：

CU C [MW]

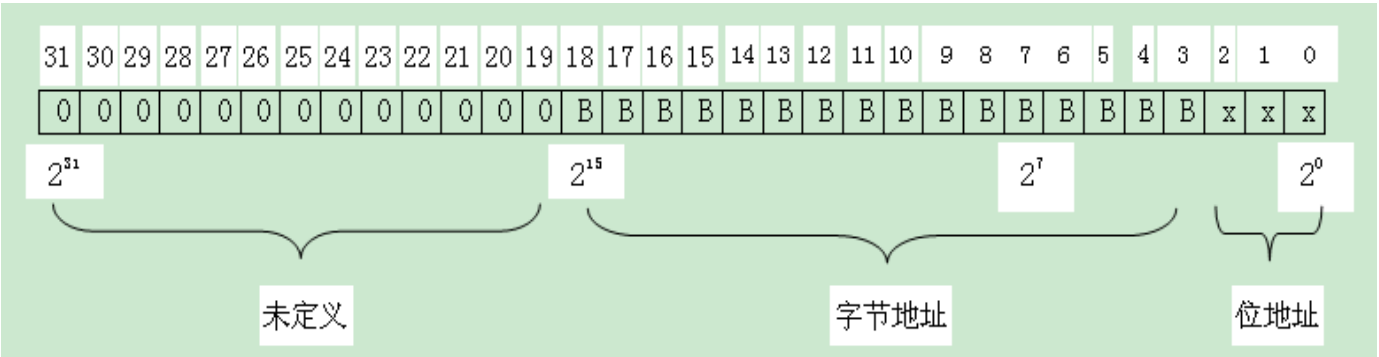
上述指令中，'C'为区域标识符，而'MW20'为一个 16 位指针。

16 位指针存储器间接寻址举例：

```
//用于定时器
L       1
T       MW0                 //将 1 传送到 MW0
A       I0.0                //如果 I0.0 = True
L       S5T#10S
SD      T[MW0]              //T1 开始计时
//              上述指令可等同于：
A       I0.0
L       S5T#10S
SD      T1
```

```
//用于打开 DB 块
L       20
T       LW20
OPN     DB[LW20]            //打开 DB20

//程序调用
L       2
T       LW20
UC      FC[LW20]            //调用 FC2

L       41
T       DBW30
UC      FB[DBW30]           //调用 FB41
```

*****32 位指针：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：



存储器 32 指针仅用于内部区域寻址。
寻址格式表示为：　地址存储器标识符 [32 位地址指针]

例如写入一个 M 的双字表示为：
```
T       MD [ LD0 ]
```
‘MD’为区域标识符及访问宽度，而 LD0 为一个 32 位指针。

32 位内部区域指针可用常数表示， 表示为 **P# 字节.位** 。如常数
```
P#      10.3
```
为指向第 10 个字节第 3 位的指针常数。
若把一个 32 位整型转换为字节指针常数，从上述指针格式可以看出，应要把该数左移 3 位(或是乘 8)即可。
```
如：     L       L#100       //Accu0 装入 32 位整形 100
         SLD     3           //左移 3 位
         T       LD0         //LD0 得到 P#100.0 指针常数
```

32 位存储器间接寻址举例：

```
//DB 块间接寻址举例
OPN     DB      1           //打开 DB1。
```

```
      OPN   DI    3          //打开 DB3，最多可以同时打开两个 DB 块。
      L     4                //装载 4 到累加器 1 中。
      SLD   3                //累加器 1 中数值左移 3 位。
      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。
      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。
      +I                     //相加
      L     DIW   [MD 28]    //装载 DB3.DBW40。
      -I                     //相减。
      T     DIW   2          //将运算结果传送到 DB3.DBW2 中。
      JC    m1

      //M 存储器连续区域操作
      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。
      ==I                    //如果数值相等跳到 m1。
      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#2.0
      /D
      +     1
      T     MD    8
m2:   NOP   0
```

10.2.2 寄存器间接寻址

通过 CPU 的地址寄存器 AR1 和 AR2 寻址方式称为寄存器间接寻址，分为内部区域间接寻址和交叉区域寻址。使用寄存器间接寻址方式的程序语句包含以下部分：

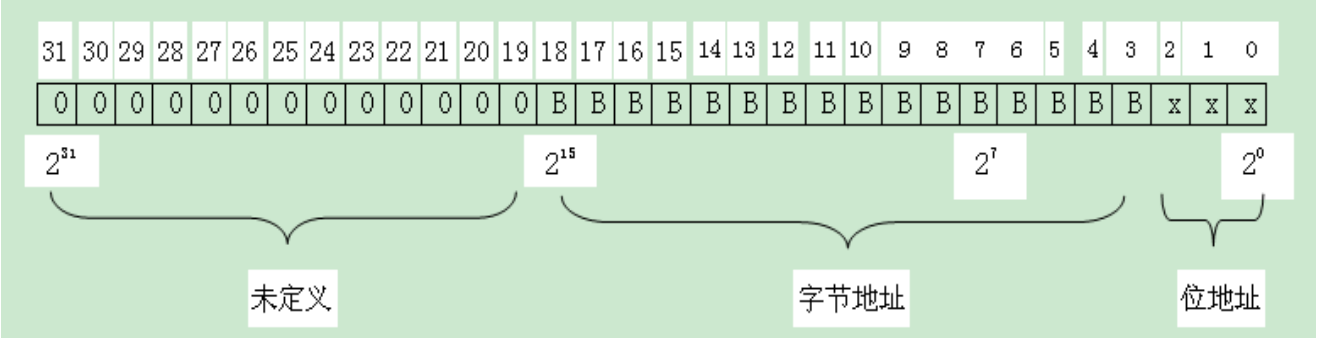
指令，地址标识符，地址寄存器标识符，偏移量

AR1，AR2 均为 32 位寄存器，寄存器间接寻址只使用 32 位指针。
与 ARx 相关的指令有：

LAR1, LAR2, TAR1, TAR2, +AR1, +AR2, LAR1 AR2, CAR 等。

以上指令使用请参考手册：开始 -> (所有)程序 -> SIMATIC -> Documentation -> English -> STEP 7 – Statement List for S7-300 and S7-400。
或点击下载该手册中文版

*内部区域寄存器间接寻址
指针格式与存储器间接寻址的 32 位指针相同，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：



该间接寻址表示为：存储器标识符 [ARx, 地址偏移量]
如：

```
L       MW      [AR1, P#2.0]
```

‘MW’为被访问的存储器及访问宽度，’AR1’为地址寄存器 1，P#2.0 为地址偏移量。

内部区域寄存寻址举例：

//DB 块寄存器内部寻址

```
OPN     DB      1               //打开 DB1。
LAR1    P#10.0                  //将指针 P#10.0 装载到地址寄存器 1 中。
L       DBW     [AR1, P#12.0]   //将 DBW22 装载到累加器 1 中。
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。
+I
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。
```

//DI、DO 区寄存器内部寻址

```
L       P#8.7                   //装载指向第 8 字节第 7 位的指针值到累加器 1
LAR1                            //累加器 1 中的指针装载到 AR1
A       I       [AR1, P#0.0]    //查询 I8.7 的信号状态
=       Q       [AR1, P#1.1]    //给输出位 Q10.0 赋值
```

*****交叉区域寄存器间接寻址

包含有存储器区域信息的指针，称为交叉区域指针。

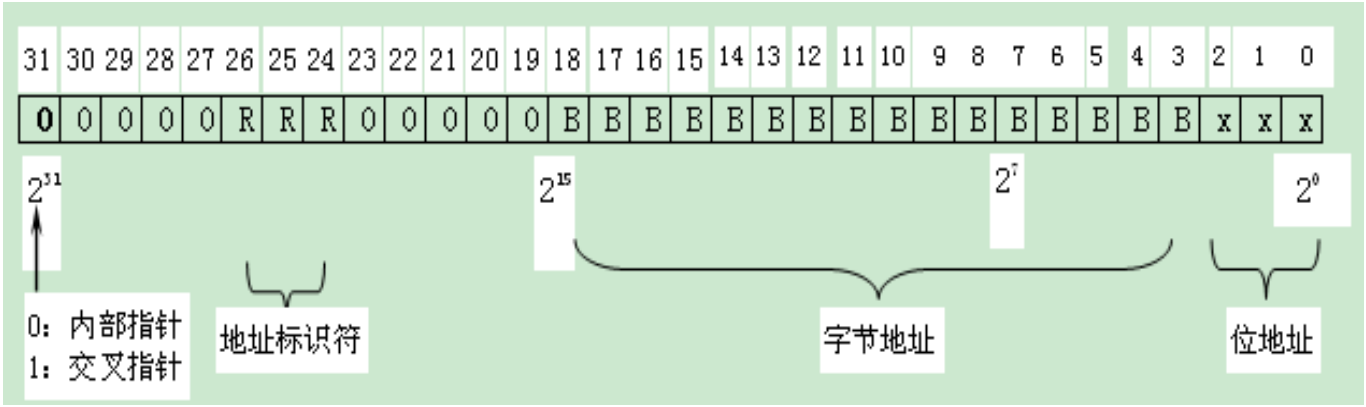
同样，交叉区域指针为 32 位，寄存器间接寻址要使用地址寄存器 AR1 或 AR2。

32 位交叉区域指针，左起 0～18 位格式与 32 位内部区域指针相同，19～23 位，27～20 位未定义，31 位为交叉区域指针标识位。

24～26 位是存储区域地址标识，8 中组合表示 8 种存储器区域：

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；

交叉区域指针格式如下：



交叉区域指针常数表达为：**P#　存储器　字节 . 位**

例如：　P#Q10.5　　//指向 Q 区第 10 字节第 5 位的指针常
　　P#DB1.DBX32.0　　//指向 DB1 区域的第 32 字节第 0 位的指针常数

交叉区域寻址表示为：　　访问宽度 [ARx, 偏移量]
例如：　L　W [AR2, P#1.0]
　　'W'为访问宽度，AR2 为地址寄存器 2，P#1.0 为偏移量。

交叉区域间接寻址举例：

```
//M 存储区
L     P#M20.0
TAR1
L     1234
T     W [ AR1, P#2.0]

//I 存储区
L     P#I0.0
```

```
LAR2
L       W[ AR2, P#0.0 ]
T       MW0
```

10.2.3 FB/FC 的指针参数传递

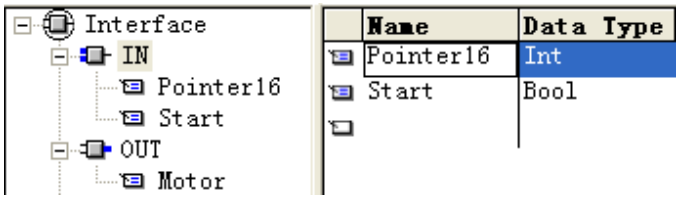
在使用 FB/FC 形参传递指针参数时，16 位、32 位指针是可用的，但 POINTER 与 ANY 指针类型也是常见的类型，因为更方便。

*16 指针用于参数传递

例如：

//编写一个 FC，作用是启动条件满足后延时 3 秒输出闭合信号

//定义 FC 的形参如下：



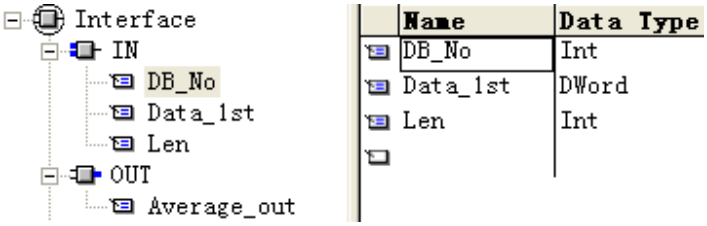
//程序如下：

```
L    #Pointer16
T    LW    0          //将 16 位指针装入 LW0
A    #Start           //Start 参数 = True 时
L    S5T#3S           //计时 3 秒
SD   T [LW 0]
A    T [LW 0]         //计时器计时到
=    #Motor           //输出 Motor = True
```

*32 指针用于参数传递

//编写一个 FC，作用是将输入 DB 块指定的区域 (实数) 求出平均值

//定义形参如下：



//程序如下：

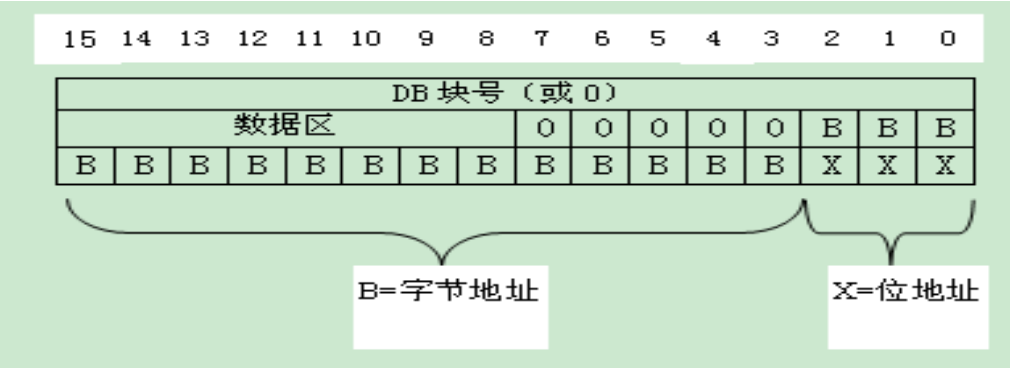
```
L    #DB_No
T    LW    0          //装载 DB 块号至 LW0
OPN  DB [LW 0]        //并打开该 DB 块

L    #Data_1st
T    LD    2          //装载第一个要计算的实数的 32 指针至 LD0
```

```
        L    0
        T    LD    8              //将'和'初始为 0

        L    #Len                 //长度
NEXT:   T    LW    6              //实数的个数装载至 LW6, 并且进入一个 LOOP 循环
        L    DBD [LD 2]           //读取 LD2 指针位置的实数
        L    LD    8
        +R                        //与'和'相加
        T    LD    8              //结果存到'和'中
        L    LD    2              //装入指针
        L    P#4.0
        +D                        //指针加 4 个字节
        T    LD    2              //结果仍存入 LD2,此时 LD2 指针指向下一个实数

        L    LW    6              //循环计数
        LOOP  NEXT                //LOOP 循环的结束

        L    #Len                 //将实数个数由 INT 转成 REAL
        ITD
        DTR
        T    LD   12
        L    LD    8              //'和'除以实数格式的实数个数
        L    LD   12
        /R
        T    #Average_out         //得到平均值，通过 Average_out 输出
```

*****POINTER 数据类型及参数传递

POINTER 是一种用于传递指针的形参数据类型，长度为 6 个字节。用于向被调用的函数 FC 及函数块 FB 传递复合数据类型（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。

格式如下：



POINTER 参数中，数据区含义如下：

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

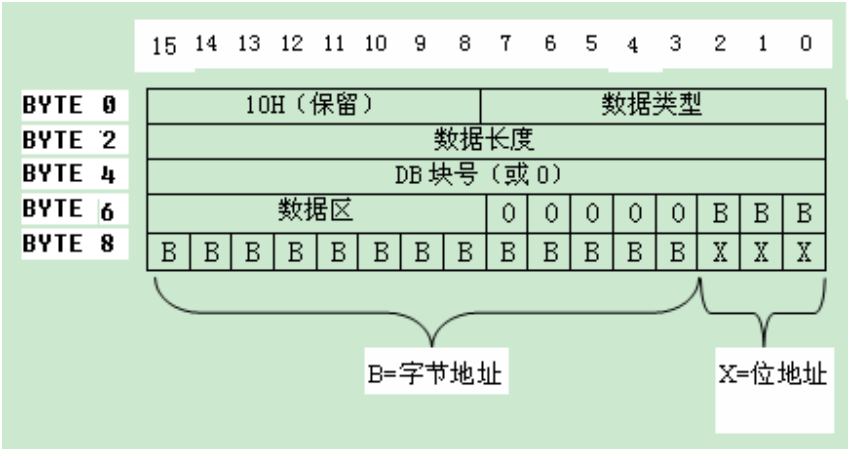
若数据区为 B#16#84,那么表示该 POINTER 指针是一个 DB 块地址，DB 块号区域应当是所指向的 DB 块号(INT 类型)并且不为零。

（请参考 ANY 数据类型举例）

*****ANY 数据类型及参数传递

POINTER 是一种用于传递指针的形参数据类型，可视为 POINTER 类型的扩展，较 POINTER 类型为复杂，长度为 10 个字节，增加的 2 字节，最高字节 (Byte 0) 固定为 B#16#10，第二字节 (Byte 1) 为 ANY 指针所指向区域的数据类型，而接下来的 2 字节 (BYTE 3, 4) 组合为一个 INT，为 ANY 指针所指定区域的长度，称为重复系数 (Repetition factor) 。其余 6 字节作用与 POINTER 类型相同。

格式如下：

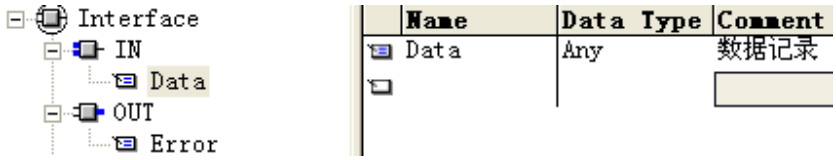


其中数据类型含义为：

数据类型代码		
十六进制代码	数据类型	简单描述
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
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
B#16#09	DATE	IEC 日期

B#16#0A	TIME_OF_DAY（TOD）	24 小时时间
B#16#0B	TIME	IEC 时间
B#16#0C	S5TIME	SIMATIC 时间
B#16#0E	DATE_AND_TIME（DT）	时钟
B#16#13	STRING	字符串
B#16#17	BLOCK_FB	FB 号
B#16#18	BLOCK_FC	FC 号
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

编程举例：
//冒泡排序程序，算法原理请参考相关资料
//此例程仅提供存于 DB 块中的 INT 类型数据排序
//结果为 INT 数据由小到大排列，保存于原 DB 块中
//FC 块，形参定义如下：



//输入参数 Data(Any 类型)；输出参数 Error(INT 类型)
//若输出参数 Error 不为 0, 则说明排序未进行，
//Error = 1, Data(ANY 类型)输入指针无效
//Error = 2, Data(ANY 类型)输入指针地址非 DB 地址
//Error = 3, Data(ANY 类型)输入指针指定区域类型非 INT 类型

```
      SET
      SAVE
      L     0
      T     #Error

//将输入 ANY 指针'Data_In'复制到 LB0~LB9
      L     P##Data
      LAR1
      L     D [AR1,P#0.0]
      T     LD     0
      L     D [AR1,P#4.0]
      T     LD     4
```

```
      L     W [AR1,P#8.0]
      T     LW     8

//ANY 指针 BYTE0 是 B#16#10
      L     LB     0
      L     B#16#10
      ==I
      JCN   ERR1

//输入数据区是否为 DB 块
      L     LB     6
      L     B#16#84
      ==I
      JCN   ERR2

//类型为 INT
      L     LB     1
      L     B#16#5
      ==I
      JCN   ERR3

//打开输入 DB 块
      OPN   DB [LW 4]

//数据起始地址去掉数据区标识部分
      L     LD     6
      L     DW#16#FFFFFF
      AD
      T     LD    10
//计算最后一个存储单元指针保存至 LD10
      L     LW     2
      L     2
      *I
      T     LD    14
      L     L#2
      -D
      SLD   3
      L     LD    10
      +D
      T     LD    10

//外循环计数 LW20，循环次数为(数据个数-1)次
      L     LW     2
      L     1
      -I
NXT2: T     LW    20
      L     LD    10
```

```
      LAR1
      L     LW    20

//嵌套循环计数 LW18，循环次数为(LW20)次
NXT1: T     LW    18
      TAR1
      L     P#2.0
      -D
      LAR1

//后一单元数据小于前一单元数据？
      L     DBW [AR1,P#2.0]
      L     DBW [AR1,P#0.0]
      <I
      JCN   L1

//否，交换 2 单元数据
      L     DBW [AR1,P#2.0]
      L     DBW [AR1,P#0.0]
      T     DBW [AR1,P#2.0]
      POP
      T     DBW [AR1,P#0.0]

L1:   L     LW    18
      LOOP  NXT1
      L     LW    20
      LOOP  NXT2
      JU    EXIT
//错误码 1，ANY 指针有错
ERR1: L     1
      T     #Error
      JU    EXIT

//错误码 2，输入数据区不是 DB 块
ERR2: L     2
      T     #Error
      JU    EXIT

//错误码 3，输入数据类型不是 INT
ERR3: L     3
      T     #Error

EXIT: NOP   0
```

在 **OB1** 程序中调用举例：

```
      A     M      0.0
```

```
        FP    M      0.1
        JCN   EXIT

        CALL  FC     3               //FC3 为上述排序程序
           Data  :=P#DB3.DBX 0.0 INT 64     //参数 Data, DB3 中 64 个 INT 排序
          Error  :=MW2

EXIT: NOP   0
```

10.3 S7-300/400 寻址方式图解