西门子ET-200PLC代理商6ES7193-0CB30-0XA0

产品名称	西门子ET-200PLC代理商6ES7193-0CB30-0XA0
公司名称	湖南西控自动化设备有限公司
价格	.00/件
规格参数	西门子:授权代理商 模块:一级代理商 德国:售后保障服务
公司地址	中国（湖南）自由贸易试验区长沙片区开元东路1306号开阳智能制造产业园（一期）4#栋301
联系电话	17838383235 17838383235

产品详情

PLC中指针和间接寻址的深度应用

在PLC（西门子）中通过使用间接寻址方法，指令所使用的地址可以改变为指向任意数量的位置。在这种情况下，一个内存位置存储指向另一个内存位置的“ 指针 ”。虽然这可能会增加故障排除的难度，但其优点是大大减少控制过程所需的网络和指令的数量。也是使用西门子提供的一些库和系统函数调用必须了解的方法。

POINTER（指针） 和任何数据类型

POINTER 数据类型用于格式化要接受为地址而不是值的数字。指针前面始终带有 P#
符号。指针地址可以采用三种不同的格式。 ANY
数据类型用于传递未知或未定义数据类型的参数。库中的某些函数使用 ANY
数据类型来处理整个内存部分。为此，使用后一个指针方法来描述一个区域。例如地址 P#DB25.DBX 0.0 Byte 14 指向 DB25 的个字节，长度为 14 个字节。

注意：只需将双字左移 3 位即可将 DINT 转换为 POINTER。

数据块指令

使用间接寻址时，有时需要首先打开一个 DB，然后开始处理地址，而不直接引用任何一个 DB。这是使用OPN指令完成的。OPN 指令可以打开共享数据块 (DB) 或背景数据块 (DI)。OPN DB 10 //开DB10作为共享数据块 L DBW 36 //将DB10的数据字36加载到ACCU1中
T MW 22 //将ACCU1的内容传送到MW22中
OPN DI 20 //打开DB20作为背景数据块 L DIB 12 //将数据字节12从DB20加载到ACCU1中
T DBB 37 //将ACCU1的内容传输到开放共享数据块DB10的数据字节37在 STL 中监视时，共享 DB 编号显示在 DB1 列中，背景 DB 编号显示在 DB2 列中。 此外，还有指令确认打开了正确的 DB 编号，并且它足够大以进行下一步操作。L DBNO //将打开的共享数据块的编号加载到ACCU1
L DBLG //将打开的共享数据块的长度加载到ACCU1
L DINO //将打开的实例数据块的编号加载到ACCU1
L DILG //将打开的实例数据块的长度加载到ACCU1

内存间接寻址

种间接寻址方法称为内存间接寻址，因为它允许一个内存位置（M、DB 或 L）确定或指向另一个位置。 存储区标识符 T、C、DB、DI、FB 和 FC 使用整数格式的字（16 位）指针位置。两个例子如下： L 5 //用指针值加载ACCU1
T MW 2 //将指针传输到MW2 L T [MW 2] //用T5的当前时间值加载ACCU1
OPN DB [#DB_Temp] //打开数据块号来自接口临时参数的DB，命名为DB_Temp存储区标识符 I、Q、M、L、DB 使用 POINTER 数据类型的双字（32 位）位置。存储区标识符 I、Q、M、L、DB 使用 POINTER 数据类型的双字（32 位）位置。 L P#0.7 //用指针值加载ACCU1
T MD 2 //将指针传输到MD2 A I [MD 2] //检查I0.7的状态
= M [MD 2] //将RLO的值分配给M0.7 OPN DB 5 //打开DB5
L P#2.0 //将指针加载到ACCU1 T #TempPointer //将指针传输到临时位置
L DBW [#TempPointer] //将DB5.DBW2处的值加载到ACCU1 L 0 //将零加载到ACCU1 >D //检查值是否大于零当监视内存间接寻址时，间接列显示指令正在使用的当前地址。 请注意，可以使用双精度数学指令对 POINTER 数据类型进行数学运算（例如 P#2.0 + P#5.0 = P#7.0）。 L P#2.0 //用指针值加载ACCU1 L P#5.0 //用第二个指针值加载ACCU1 +D
T MD 0 //现在MD0包含值P#7.0由于Bit进位是8，因此结果为 P#8.7 + P#1.1 = P#10.0，而不是 P#9.8。这些方法可用于在循环中偏移地址或增加/减少指针。

地址寄存器

除了常规累加器外，还有两个 32 位地址寄存器（AR1、AR2）用于存储寄存器间接寻址方法中使用的指针。一系列不同的加载和传输类型指令可用于与 AR1 配合使用。AR2 也有类似的套件。 可以通过以下方式直接在 AR1 和 AR2 上完成加法：

区域内部寄存器间接寻址

区域内部寄存器间接寻址方法使用地址寄存器之一加上指针来确定指令要引用的地址。格式为： 地址标识符[地址寄存器，指针] 地址标识符可以是位、字节、字或双字形式的 I、Q、M、L、DI 或 DB。地址寄存器必须预先加载双字指针，而不参考地址标识符。准确的地址是通过地址寄存器与指针相加来确定的。下面的示例显示了使用位位置的区域内部方法。 L P#0.7 //用指针值加载ACCU1
LAR1 //将指针加载到AR1中 A I [AR1, P#0.0] //检查输入I0.7
= Q [AR1, P#1.1] //如果RLO=1，则打开Q2.0

跨区域寄存器间接寻址

跨区域寄存器间接寻址与区域内部方法类似，只是加载到地址寄存器中的指针引用内存区域（例如 P# M 10.0 或 P# DBX 0.0）。这意味着如果引用位，则不需要在左括号之前使用地址标识符，否则它将是 B 代表字节、W 代表字或 D
代表双精度。下面的示例显示了使用位位置的区域交叉方法。L P#I0.7　//用指针值加载ACCU1
LAR1　//将指针加载到AR1中 L P#Q124.0　//用指针值加载ACCU1
LAR2　//将指针加载到AR2中 A [AR1, P#0.0]　//检查输入I0.7
= [AR2, P#1.1]　//如果RLO=1，则打开Q125.1

下一个示例展示了使用字和双字格式的区域交叉方法。

L P#M0.0　//用指针值加载ACCU1 LAR1　//将指针加载到AR1中
L W [AR1, P#10.0] //将地址由AR1内容加上10字节（MW10）确定的字加载到ACCU1
OPN DB 5　//打开DB5 L P#DBX 0.0　//将指针值加载到ACCU1
LAR2　//将ACCU1中的指针加载到AR2 L L#0　//将零加载到ACCU1
T D [AR2, P#50.0] //将ACCU1中的值传输到确切位置是AR2地址
//加上50字节的双字中(DB5.DBD50)