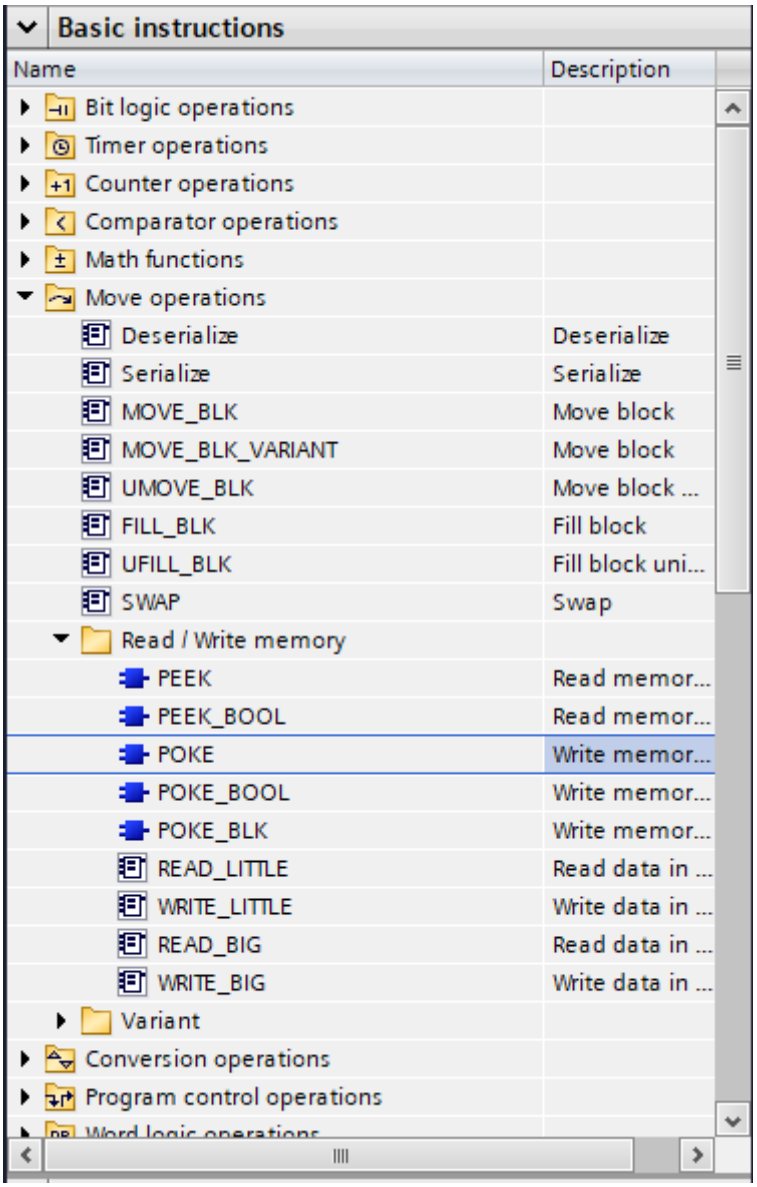
作者：北岛李工

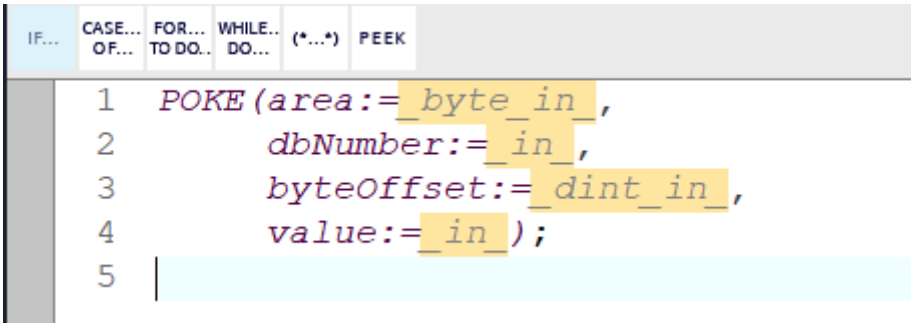
上一篇文章我们介绍了PEEK指令，它是用来读取数据的指令。除了数据的读取，SCL还提供了数据写入的指令——POKE。今天这篇文章，我们来学习下POKE指令的用法。

POKE指令用来将某一个存储区地址的数据写入到另一个存储区地址，无须指定数据类型。

可以在【基本指令(Basic instructions)】-【移动操作(move operations)】-【读写内存(Read/Write memory)】中找到该指令，如下图：



将POKE指令添加到程序块中的初始状态如下：



可以看到，POKE指令有四个参数：area ,dbNumber,byteOffset和value，各参数的含义如下：

1、area：字节型数据（Byte），用来指定访问存储区的类型。其取值包括如下内容：16#81表示输入缓存区（I）、16#82表示输出缓存区（Q）、16#83表示位存储区（M）、16#84表示数据块（DB）、16#1表示外设输入（PI）。其

中，16#84只能访问“标准的”数据块；16#1对外设的读取，只能在S7-1500系列PLC中使用。

2、dbNumber：双整型数据（DINT），用来指定数据块的编号，仅在访问数据块时使用；访问其它存储区时设置为0；

3、byteOffse：双整型数据（DINT），用来指定写入数据的地址偏移量；

4、value：可以为字节型、整型、双整型数据，用来表示要写入的数据值及类型。必须为变量，不能为常量。POKE指令根据value的数据类型来决定写入多少个字节。

举个例子：

使用POKE指令将位存储区MB100的值写入到输出缓存区QB10，SCL代码如下：

```
//SCL POKE指令使用例程
//www.founderchip.com
//将MB100的值写入到QB10
POKE(area:=16#82,//输出缓存区
     dbNumber:=0,
     byteOffset:=10,
     value:="Byte_M100");//MB100

```

如果是操作整型或字类型的数据，只需要改变value的数据类型。例如下面的代码将MW102的值写入到输出缓存区QW12：

```
//SCL POKE指令使用例程（整型/字操作）
//www.founderchip.com
//将MW102的值写入到QW12
POKE(area := 16#82,
     dbNumber := 0,
     byteOffset := 12,
     value := "Word_M102");

```

同样的道理，下面的代码将MD90的值写入到DB5.DBD10中：

```
//SCL POKE指令使用例程（双整型/双字操作）
//www.founderchip.com
//将MD90的值写入到DB5.DBD10
POKE(area := 16#84,
     dbNumber := 5,
     byteOffset := 10,
     value := "DWord_M90");

```

如果要操作布尔型数据，则需要使用POKE_BOOL指令。从指令列表中添加该指令的初始状态如下：

```
26  POKE_BOOL(area:=_byte_in_,
27            dbNumber:=_in_,
28            byteOffset:=_dint_in_,
29            bitOffset:=_int_in_,
30            value:=_bool_in_);
31
```

该指令有五个参数：area ,dbNumber,byteOffset,bitOffset和value。其中：

1、area,dbNumber,byteOffset与POKE指令相同；

2、bitOffset：整型数据（INT），用来指定要写入的位的偏移；

3、value：要写入的地址或布尔数据常数；

举个例子：将M0.0的值写入到Q1.5，可以使用下面的代码：

```
26  //SCL POKE指令使用例程（布尔型数据操作）
27  //www.founderchip.com
28  //将M0.0的值写入到Q1.5
29 POKE_BOOL(area:=16#82,
30            dbNumber:=0,
31            byteOffset:=1,
32            bitOffset:=5,
33            value:="Bit_M00");//bit M0.0
34
```

除了POKE和POKE_BOOL，SCL语言还提供POKE_BLK用来进行较大数据的移动与拷贝。名称中的“BLK“为Block的缩写，即数据块的意思。

从指令列表中添加POKE_BLK的初始状态如下：

```
36  POKE_BLK(area_src:=_byte_in_,
37           dbNumber_src:=_in_,
38           byteOffset_src:=_dint_in_,
39           area_dest:=_byte_in_,
40           dbNumber_dest:=_in_,
41           byteOffset_dest:=_dint_in_,
42           count:=_dint_in_);
43
```

可以看到，该指令有7个参数，其中：

1、area_src：字节型数据（Byte），用来指定源数据存储区。其取值包括如下内容：16#81表示输入缓存区（I）、16#82表示输出缓存区（Q）、16#83表

示位存储区（M）、16#84表示数据块（DB）；

2、dbNumber_src：双整型数据（DINT），用来指定源数据块的编号，仅在访问数据块时使用，访问其它存储区时设置为0；

3、byteOffset_src：双整型数据（DINT），用来指定源数据存储区中写入数据的地址偏移量；

4、area_dest：字节型数据（Byte），用来指定目标数据存储区。其取值包括如下内容：16#81表示输入缓存区（I）、16#82表示输出缓存区（Q）、16#83表示位存储区（M）、16#84表示数据块（DB）；

5、dbNumber_dest：双整型数据（DINT），用来指定目标数据块的编号，仅在访问数据块时使用，访问其它存储区时设置为0；

6、byteOffset_dest：双整型数据（DINT），用来指定目标数据存储区中写入数据的地址偏移量；

7、count：双整型数据（DINT），用来指定需要拷贝的字节数；

举个例子：

将DB100.DBB0开始的20个字节拷贝到DB102.DBB40开始的20个字节，代码如下：

```
36  //SCL POKE_BLK指令使用例程
37  //www.founderchip.com
38  //将DB100.DBB0开始的20个字节拷贝到DB102.DBB40开始的20个字节
39 POKE_BLK(area_src:=16#84,
40           dbNumber_src:=100,
41           byteOffset_src:=0,
42           area_dest:=16#84,
43           dbNumber_dest:=102,
44           byteOffset_dest:=40,
45           count:=20);
46
```

好了，关于POKE指令就先介绍到这里。欢迎扫描下方的二维码关注我们的微信公众号。


方正智芯
Founder Chip
长按扫码关注我们
方正智芯
公众号：founderchip
官方网站：www.founderchip.com
原创工业智能控制领域（PLC、单片机、通信）的技术分享