FC 参数处理过程解析：

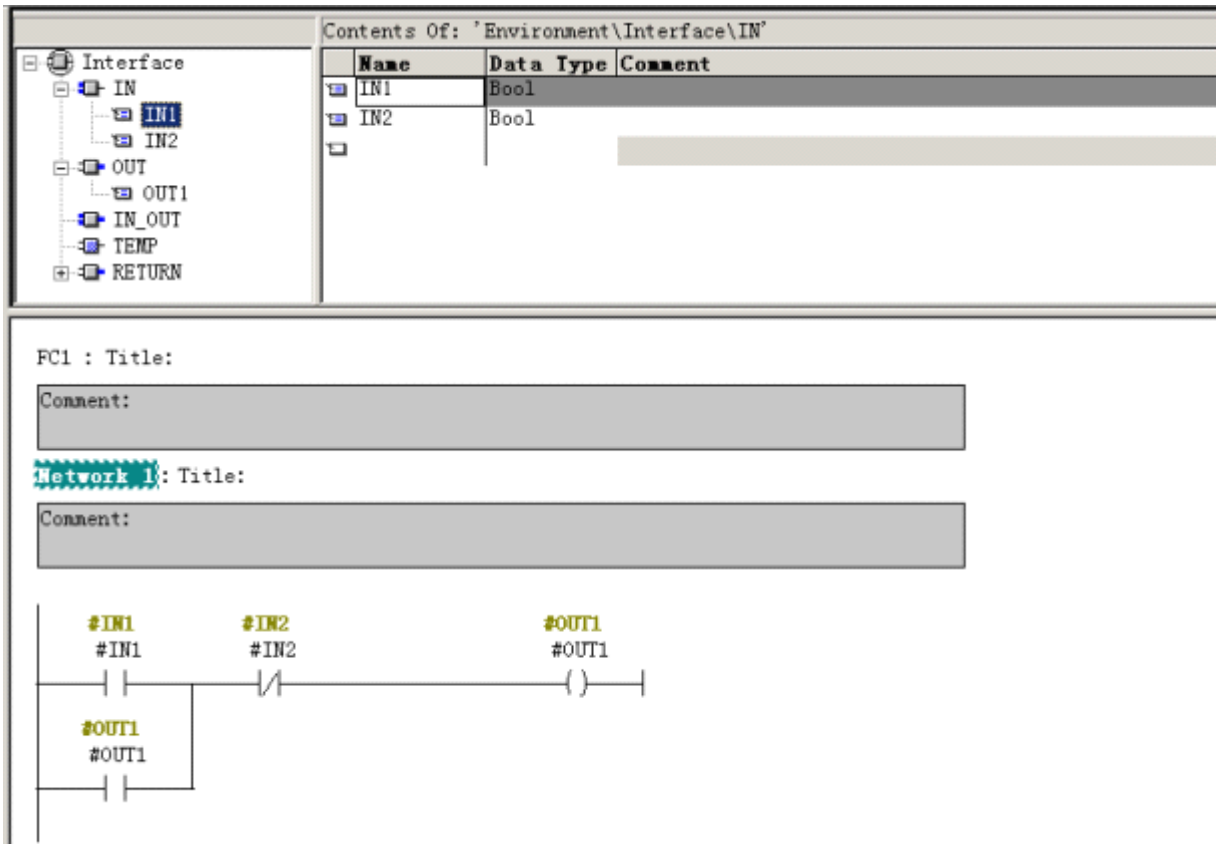
本例解决了在多次调用 FC 时，输出线圈错误的问题。

实例：

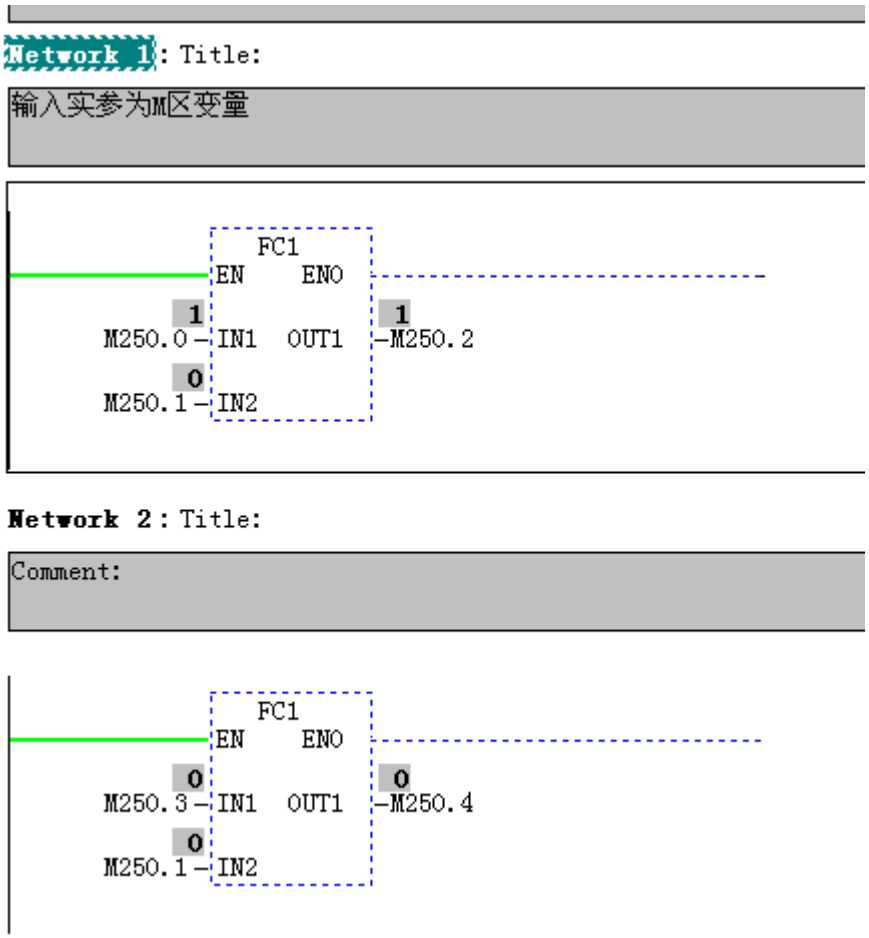
新建一个 FC 块，编写一个简单的按钮启停程序,如下图所示：

2 个 bool 型输入变量 IN1,IN2;一个 bool 型输出变量 OUT1;

以及完成简单的按钮启停程序如下：

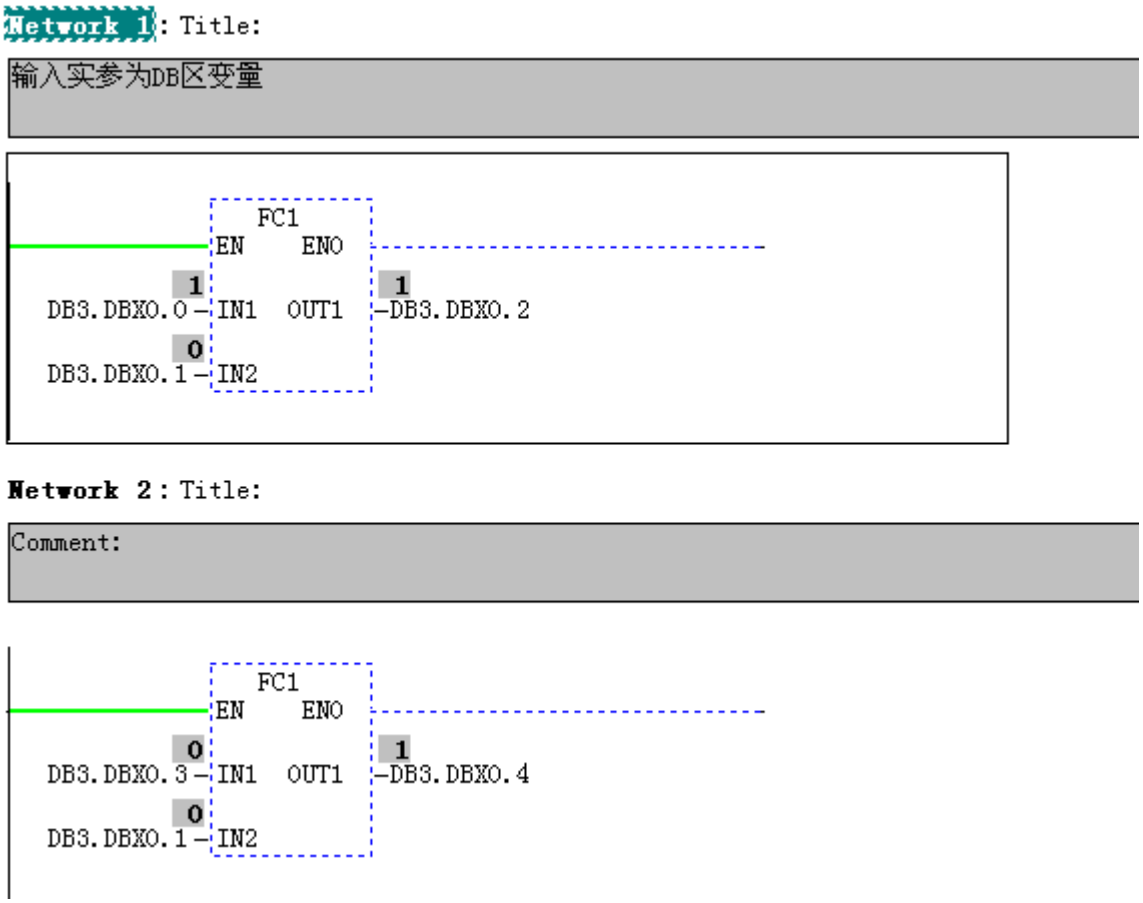


1. 在 OB1 中调用 FC1,实参输入为 M 区变量，观察结果如下：



如图：M250.0 有输入时，M250.2 输出为 1；M250.3 没有输入，M250.4 输出为 0，结果符合程序预期，运行正确；

2. 在 OB1 中再次调用 FC1，赋给实参为 DB 区的变量，如下图所示：



注意，如上图：但把 DB3.DBX0.0 置 1 时，DB3.DBX0.2 有输出；但同时发现，Network 2 中，DB3.DBX0.4 也输出了 1，这是为什么呢，本来不应该有输出的啊？

☺ 问题提出：为什么同样的 FC,当参数为 M 区变量和 DB 区变量时，执行的结果会不同呢，FC 对于参数的调用机制是怎么样的？

问题解答：

在 STEP 7 V5.4 编程手册中，我们可以找到这样的描述：

在编程调用块时，请确保写入声明为 OUTPUT 的参数。否则，输出值为随机值！对于功能块，该值为最后一个调用指定的实例 DB 中的数值；对于功能，该值为位于 L 堆栈的数值。注意以下几点：

- 调用 FC 时，初始化所有 OUTPUT 参数；
- 尽量不要使用任何置位和复位指令，这些指令取决于 RLO,当 RLO 的值为 0 时，将保持随机值；
- 如果在块内跳转，那么确保没有跳过写入 OUTPUT 参数的位置。

结论：FC 调用输出实参为数据块时的处理，与输出实参为 M、I、Q、PQ 不同。

在输出实参为 M、I、Q、PQ 时，FC 输出形参（指针）指向实参实际存储地址，如果在 FC 中，没有对输出形参形成实质性操作，则形参指针所指向的实际存储区内容不会改变，对于参数传递过程而言，仅仅是调用时形参指向实参，调用结束时形参指针区域释放的一个过程。

在输出实参为 DB 块时，FC 开始调用时，先在 V 区为该形参分配一个存储区（可能是位，也可能是字节或其它，根据形参参数类型而定），每一次调用 FC，根据调用 FC 时所传递的实参类型不同，为输出形参分配的 V 区区域可能不同，比如一个 FC 有一个位输入形参，一个位输出形参，在第一次调用时，实参分别是 I0.0、DB0.DBX0.0，则为输出形参分配的 V 区为 V20.1；第二次调用 FC，实参分别为 DB0.DBX0.1、DB0.DBX0.2，则为输出形参分配的 V 区为 V20.2，原来的 V20.1 被输入形参所对应的实参 DB0.DBX0.1 内容所填充。如果第二次调用 FC 时，实参为 M0.1、DB0.DBX0.2，则为输出形参分配的 V 区仍为 V20.1。

其后，FC 在实际运行时，针对输出形参（数据块）的处理其实都是在对该 V 区进行处理，在 FC 结束调用后，不管在 FC 中有没有对该 V 区进行实质性处理，该 V 区的内容均会被拷贝至另一个存储区，比如 L 区，该 L 区与实参区对应，但不在同一区，此时并一直等到本次 OB 循环结束，该 L 区内容都不会被拷贝至实参区，所以这个时候实参区的内容还不会有变化。

等到下一个 OB 循环开始之前，该 L 区内容就会被拷贝至实参区。

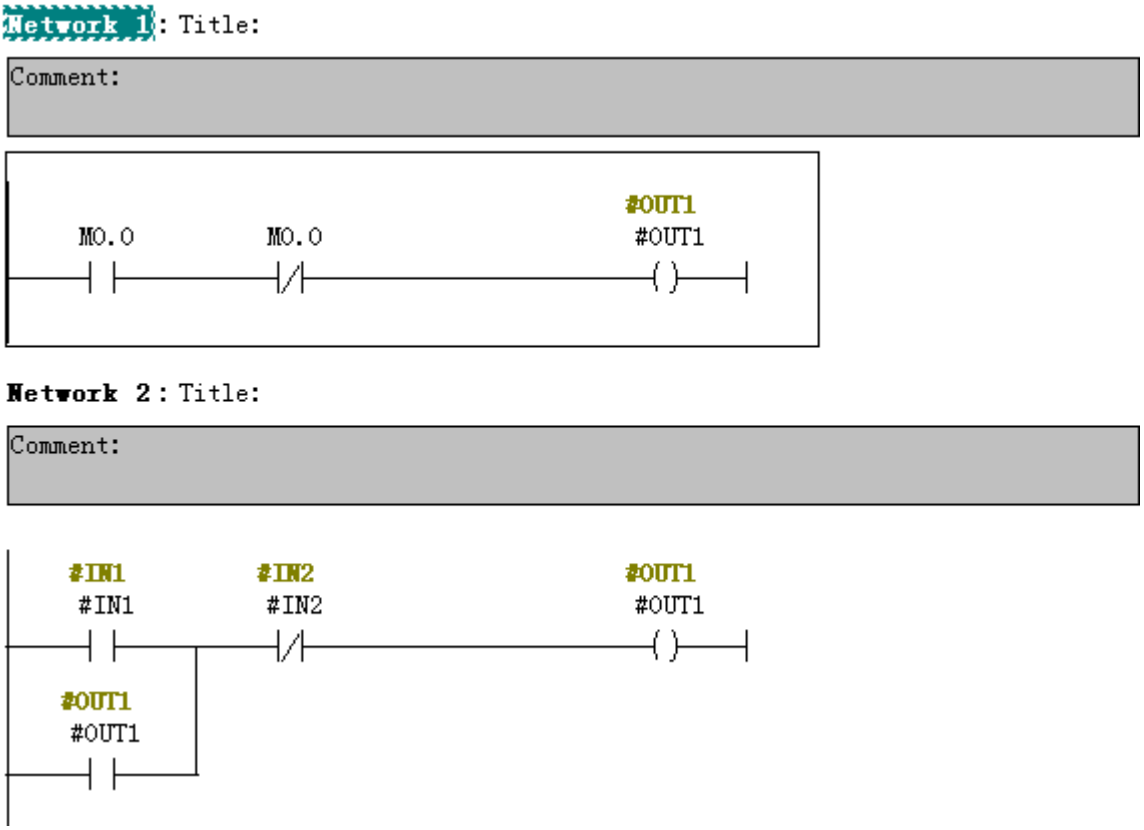
根据以上内容，结合本例分析：

1：第一次调用 FC1 时，在 V 区为形参分配一个存储区，本例中实参为 DB3.DBX0.0，DB3.DBX0.1，输出实参为 DB3.DBX0.2，则为形参分配的 V 区为 V20.1,V20.2,V20.3;程序执行完毕后，V20.3 值为 1，并输出给 DB3.DBX0.2;

2:　第二次调用 FC1 时，同样为形参分配的 V 区为 V20.1,V20.2,V20.3,但程序并未对 OUT1 变量进行初始化操作，此时 V20.3 的值仍然为 1，即 OUT1 保持为第一次调用后的 V20.3 的值，所以第二次调用 FC1 时，输出 DB3.DBX0.4 也变为了 1！

解决办法：

1. 在 FC1 中对 OUT1 进行初始化操作，如下图所示：



2. 把 OUT1 变量改为 IN/OUT 类型。
3. 以上 2 点适用于所有 FC 调用，切记。